# Diseño y Desarrollo de Personajes con Presencia Social en Videojuegos de Realidad Virtual


**Trabajo de Fin de Grado**
**Curso 2019–2020**

**Autores**
**Adriana del Castillo Espejo-Saavedra**
**Raúl Serrano Gómez**

**Director**
**Prof. Dr. Federico Peinado Gil**
**Codirector: Dr. Manuel López Ibáñez**

**Grado en Desarrollo de Videojuegos**
**Facultad de Informática**
**Universidad Complutense de Madrid**

# Resumen

El año 2016 supuso la reaparición de una ola de productos de Realidad Virtual a la que se unirían grandes empresas como Facebook, Sony o HTC y que pondría énfasis en las aplicaciones para la industria del entretenimiento. Desde entonces se suceden los avances tecnológicos, de accesibilidad y de jugabilidad que tratan de ser explotados en el mundo del videojuego comercial.

Las nuevas tecnologías permiten hoy alcanzar un nivel de inmersión aún mayor que el de los videojuegos convencionales. Para aprovecharlo, debemos mantener la sensación de presencia durante toda la experiencia. En particular, algo que el jugador desea es poder interactuar con los personajes del juego, que sean percibidos como entes inteligentes, atentos y reactivos antes nuestras acciones y nuestra presencia, lo cual es raro de encontrar en los videojuegos actuales. Para crear este tipo de personajes es clave el uso de técnicas de Inteligencia Artificial.

El objetivo de este proyecto es construir una herramienta para Unity que facilite el diseño y desarrollo de personajes sociales mediante la definición de un conjunto de eventos básicos ante los que el personaje debe ser capaz de dar respuesta, y permitiendo su extensibilidad a todo tipo de comportamientos específicos de un título concreto.

El análisis y diseño de los objetivos propuestos concluyó en el paquete de tareas y comportamientos para personajes no jugables denominado Social Presence VR. Además, su eficacia ha sido comprobada mediante test A/B con usuarios. La herramienta es gratuita y se encuentra disponible en la plataforma GitHub.

## Palabras clave

Desarrollo de Videojuegos, Inteligencia Artificial, Herramienta de Desarrollo, Unity, Interactividad, Inmersión, Instructor.

# Índice

# Índice de figuras

Capítulo 1

# Introducción

La Realidad Virtual (RV) entró con más fuerza que nunca en la industria del videojuego a partir del año 2016 y ha supuesto uno de los cambios tecnológicos más importantes de los últimos tiempos para este tipo de entretenimiento (Mundo Virtual, 2016). Si bien el mercado ha seguido creciendo (Itreseller, 2019), es cierto que todavía hay diferentes factores que le impiden despegar y alcanzar cifras similares a las de otros productos y servicios tecnológicos.

Uno de los principales motivos de su estancamiento es el coste que estos productos tienen para el usuario final. El precio de estos dispositivos y la necesidad de conectarlos a un ordenador con hardware gráfico muy potente y actualizado, hace que sean alcanzables para muy pocos consumidores. De todas formas estos años se aprecia que el paso del tiempo y los consecuentes avances en ingeniería permiten que estos dispositivos se vuelvan más económicos y por tanto asequibles. Uno de los modelos de dispositivos de RV más accesibles para el gran público es *Oculus Quest* (Facebook Technologies, LLC, 2019), pues no sólo tiene un precio reducido (Xataka, 2020) sino que ya no necesita de un ordenador al que conectarse, lo que está permitiendo popularizar mucho las experiencias en RV.

Concretamente en España, en 2019 se ha producido un descenso en el número de aplicaciones desarrolladas para RV, según se informa en el Libro Blanco del Desarrollo Español de Videojuegos (Desarrollo Español de Videojuegos (DEV), 2019). Esto posiblemente es debido a la complejidad de su desarrollo, ya que se requieren especialistas en el tema y todavía hay escasez de formación. Además, los desarrolladores necesitan herramientas que faciliten el trabajo en este tipo de proyectos, sin necesidad de tantos conocimientos específicos.

Actualmente ya existen herramientas para entornos de desarrollo tan populares como Unity (Unity Technologies, 2005) que proporcionan funcionalidades básicas para RV, como por ejemplo *VRTK* (Extend Reality Ltd, 2018). Pero en un mercado con tanta proporción de productos que son una mera traslación de videojuegos convencionales (basados en pantallas, teclado y ratón o controlador para juegos), hacen faltan novedades que se diferencien y aprovechen de verdad las cualidades de la RV. Como decíamos, sería interesante contar con herramientas que faciliten la creación de mundos virtuales más inmersivos. Y uno de los aspectos que más afectan a la sensación de presencia del jugador son los personajes no jugadores (NPCs del inglés *Non-Player Characters*), que deben resultar creíbles, mostrarse vivos y reaccionar con naturalidad ante las acciones del avatar del jugador.

## 1.1. Realidad Virtual

Los videojuegos para RV tienen un potencial de recrear la sensación de presencia en el jugador muy difícil de conseguir en un videojuego convencional de los que se disfrutan en pantallas convencionales, con teclado y ratón, o controlador de juegos. La posibilidad de sentirse parte de un mundo virtual y ser capaces de interactuar con nuestras propias manos y con el movimiento del cuerpo sobre el escenario, hace que la presencia sea el punto más fuerte de esta nueva tecnología.

No solo se produce una mejora en la inmersión, también obtenemos una mayor libertad y expresividad en la interacción a la hora de jugar. La entrada del jugador no solo está limitada a la pulsación de botones o movimientos de joystick o del ratón. Ahora dicha entrada es un flujo continuo de datos, que nos proporcionan información constante sobre las acciones del jugador, de su cabeza y de sus manos.

Sin embargo, esta notoria capacidad de mejorar la experiencia de los jugadores es un arma de doble filo. Estar presentes en el mundo virtual y tener mayor libertad de acción hace que cualquier pequeña disonancia con respecto a lo que esperamos del mundo real tenga una mayor posibilidad de "sacarnos" de la experiencia. Diseñar mal un videojuego para RV no sólo hará que sea aburrido, sino que puede llegar a hacer que un jugador sufra algún malestar físico, como mareos, síntoma habitual de la llamada "enfermedad del simulador" o *simulator sickness* (Ward, 2018).

Desarrollar videojuegos para RV, por tanto, implica desarrollar conociendo las limitaciones, tanto tecnológicas como de diseño, o bien contar con herramientas que tengan en cuenta estas limitaciones. Exige un conocimiento preciso de cuáles son los problemas principales y cuáles son las pautas generales a seguir. Los puntos de mayor conflicto son actualmente el movimiento

del avatar del jugador, la interfaz extradiegética, la interacción con el propio entorno virtual y muy especialmente la interacción con los NPCs.

Uno de los mayores problemas de los videojuegos para RV es la herencia directa de elementos propios de videojuegos que funcionan con pantallas convencionales, teclado, ratón o controlador de juegos. Estos elementos, que pueden funcionar bien en ese tipo de juegos, no tienen porque dar los mismos resultados en uno de RV. Existe una gran cantidad de juegos *portados* muy exitosos de pantalla convencional a RV, como por ejemplo *Fallout 4 VR* (Bethesda Game Studios, 2017). Resulta muy emocionante explorar los enormes mundos abiertos de este tipo de juegos, pero luego es habitual encontrar carencias jugables que impiden alcanzar la misma sensación de presencia que podrían conseguirse si el título hubiese sido desarrollado específicamente para este plataforma, como por ejemplo *Job Simulator* (Owlchemy Labs, 2016).

## 1.2. Propósito del trabajo

La dificultad que supone desarrollar un juego para RV ha hecho que el número de aplicaciones desarrolladas haya disminuido de forma considerable en el último año. Además, según los datos obtenidos en España, actualmente existe un 2 % de expertos en RV y un 9 % de las empresas manifiestan expresamente tener dificultades para encontrar expertos en este sector. Esto supone que la demanda de expertos en RV es notablemente mayor que el número actual de expertos en RV.

El propósito de este trabajo es contribuir a que existan herramientas de desarrollo que faciliten la creación de experiencias inmersivas con personajes en videojuegos para RV. Se busca facilitar el desarrollo para equipos que no dispongan de expertos en estas tecnologías ni puedan permitirse contratarlos. Este es el perfil de nuestros potenciales usuarios.

Actualmente, ya existen herramientas de desarrollo que facilitan crear entornos virtuales para RV, proporcionando facilidades para interactuar con el entorno, para el manejo de la interfaz y para ocuparse del movimiento del avatar del jugador. Un ejemplo de este tipo de herramienta sería VRTK.

Sin embargo, en el mercado apenas existen herramientas que ayuden en el desarrollo de personajes para RV. Por este motivo hemos optado por centrarnos aquí en una herramienta que facilite la creación de NPCs que reaccionen de manera natural a las acciones del avatar del jugador y haga que este sienta una mayor presencia social. Esta herramienta busca proporcionar los comportamientos más básicos que todo personaje debería tener para formar parte de un mundo virtual en RV, así como permitir al desarrollador extender el sistema con reacciones a eventos específicos de su proyecto.

La herramienta se constituirá como un complemento del entorno de desarrollo de videojuegos más utilizado hoy en día, *Unity*. A nivel mundial un 73 % de estudios de desarrollo de menos de 50 empleados lo utilizan y concretamente en nuestro país, un 85 % de las empresas lo utiliza frente al segundo motor más utilizado, *Unreal Engine* (Epic Games, 1998), con un 23 %, según el Libro Blanco de los Videojuegos.

Las principales funciones que buscamos con este sistema son: la detección de acciones básicas por parte del avatar del jugador a las que un NPC creíble debería responder, la construcción de algún tipo de respuesta a estas acciones básicas, y la personalización del sistema para detectar y actuar de cualquier forma deseada a nuevas acciones del jugador, siempre con el propósito de mejorar la sensación de presencia. Todo el sistema se plantea de tal manera que tenga la capacidad de ser mejorado y ampliado fácilmente por futuros desarrolladores que deseen contribuir añadiendo funcionalidad.

Para ilustrar las posibilidades de la herramienta, una vez desarrollada esta, crearemos también un prototipo de juego utilizando el sistema propuesto. Además, realizaremos un test A/B con usuarios reales, valorando la experiencia y la sensación de presencia del mismo prototipo usando, por un lado un NPC construido con nuestra herramienta y por otro un NPC que no ha sido diseñador específicamente para suscitar presencia social en RV.

## 1.3. Asignaturas relacionadas

A continuación hacemos una descripción de las relaciones de este Trabajo de Fin de Grado (TFG) con respecto a las diferentes áreas de los estudios de grado, relacionándolo con algunas de las asignaturas cursadas.

La utilización de diferentes algoritmos de Inteligencia Artificial(IA) para la técnica de movimiento o para la toma de decisiones, hace que la asignatura de *Inteligencia Artificial para Videojuegos* sea la más estrechamente relacionada con este trabajo.

El diseño tanto de la herramienta como del prototipo de ejemplo, así como el desarrollo de documentos de diseño (GDD, del inglés *Game Design Document*) se apoya en las asignaturas de *Diseño de Videojuegos*, *Desarrollo de Sistemas Interactivos* e *Interfaces de Usuario*.

La realización de test A/B y valorar la experiencia de usuario se apoya en la asignatura de *Usabilidad y Análisis de Juegos*. La producción del proyecto, organización y gestión del repositorio y tareas se basan en *Metodologías Ágiles de Producción*. La asignatura *Motores de Videojuegos* está también relacionada con el proyecto por el uso continuado de el entorno de desarrollo de Unity.

Las asignaturas de *Sonido en Videojuegos* e *Informática Musical* han proporcionado los conocimientos para el manejo de herramientas de edición de sonido y creación de música digital, útiles para elaborar los efectos de sonido y música del prototipo realizado.

El Grado en Desarrollo de Videojuegos dispone de un listado de asignaturas relacionadas con la programación que, aunque no tengan una relación directa con la temática del trabajo, son imprescindibles, pues nos han proporcionado los conocimientos de programación y la habilidad para diseñar arquitecturas software y estructurar y resolver debidamente los problemas técnicos. Estas asignaturas son *Fundamentos de la Programación*, *Estructuras de Datos y Algoritmos*, *Tecnología de la Programación de Videojuegos*, *Informática Gráfica*, *Métodos Algorítmicos en Resoluciones de Problemas*, *Videojuegos en Consola* y *Videojuegos para Dispositivos Móviles*.

Además, las diferentes asignaturas de *Proyecto* y *Prácticas en Empresa*, aparte del desarrollo de la programación, han aportado habilidades para trabajar en equipo y desarrollar un proyecto como este, más ambicioso y en un espacio prolongado de tiempo.

## 1.4. Estructura de la memoria

La estructura de este trabajo se compone de los siguientes capítulos:

- En el Capítulo 1 hemos expuesto nuestra propuesta, presentamos las capacidades que tiene la RV, explicamos la motivación de desarrollar este proyecto y relacionamos el alcance de este trabajo con respecto a las asignaturas del grado.
- En el Capítulo 2 se expone una revisión del estado de la cuestión en los campos más relevantes para este proyecto. Se incluye un estudio de la comunicación no verbal y del proceso de toma de decisiones de la IA, un análisis de la experiencia del usuario en RV y se cierra con una recopilación de las referencias más relevantes para el desarrollo de la herramienta.
- En el Capítulo 3 se presentan los objetivos principales del trabajo y se hace una especificación de todos los requisitos necesarios para poder desarrollar la herramienta propuesta.
- En el Capítulo 4 se explica la metodología usada para desarrollar el proyecto y qué herramientas concretas se han utilizado en el proceso.
- En el Capítulo 5 se realiza una explicación detallada del análisis, diseño e implementación de nuestro proyecto, tanto de la herramienta como de la escena de ejemplo que se ha creado.

- En el Capítulo 6 se comenzará con la explicación de las pruebas y posteriormente, se comentarán los resultados obtenidos de las pruebas con usuarios reales que hemos realizado para validar el funcionamiento y evaluar la utilidad de esta herramienta, siendo el resultado en general positivo.
- En el Capítulo 7 damos una conclusión al proyecto, explorando si se han cumplido los objetivos y se plantean líneas de trabajo futuro.

Capítulo 2

# Estado de la cuestión

Nuestro proyecto se centra en la creación de una herramienta que permita construir NPCs interactivos que mejoren la presencia social en un videojuego de RV. Se pretende crear una plantilla de comportamiento donde los desarrolladores puedan modificar ciertos parámetros del NPC según les convenga.

A esta herramienta le acompañará como ejemplo una escena que contendrá un NPC que desempeñará el rol de instructor. Este asignará una tarea al jugador, le irá ayudando durante la realización de esta y le corregirá cuando sea necesario.

Para poder realizar este tipo de NPC, nuestra revisión del estado de la cuestión se centrará en los siguientes puntos:

1. **Comunicación no verbal y rol del instructor.**
2. **Toma de decisiones en los videojuegos.**
3. **Experiencia de usuario en RV**
4. **Videojuegos de referencia**

Primeramente, el proyecto se centrará en el análisis de la comunicación interpersonal, en concreto la comunicación no verbal. Esto es debido a que actualmente es tecnológicamente complejo plantear la posibilidad de una comunicación verbal recíproca entre el NPC y el jugador, por lo que no abordaremos los problemas de este tipo de comunicación. Esto puede quedar justificado desde el punto de vista de la narrativa del juego de muchas maneras, como por ejemplo con un NPC que no sea humano.

Ya que el tipo de NPC en el que nos centraremos tomará el papel de instructor, se han estudiado las cualidades necesarias que necesita una persona para poder ejercer dicho rol.

Seguidamente se estudiarán las principales técnicas de IA para videojuegos. De entre todas ellas, se profundizará en mayor medida en el funcionamiento de los árboles de comportamiento, ya que es el que mejor se adapta al tipo de toma de decisiones que se quiere implementar en los personajes.

El tercer apartado se dedicará al estudio de la experiencia de usuario en RV, analizando conceptos relevantes para el proyecto como la presencia o el peligro del *simulation sickness*, así como a describir las características primordiales de todo videojuego que desee garantizar una buena experiencia en RV.

A continuación, el último apartado irá enfocado en la búsqueda y el estudio de videojuegos o herramientas de RV que hayan desarrollado NPCs interactivos y que nos puedan servir, por tanto, de referente.

## 2.1. Comunicación no verbal y rol del instructor

Como ya se ha explicado, este apartado irá enfocado principalmente al estudio de la comunicación no verbal y concretamente sobre el rol y tipo de interacción que tiene un instructor. Consideramos relevante un estudio previo de la comunicación no-verbal entre personas para después definir dicho comportamiento mediante técnicas de IA.

### 2.1.1. Comunicación no verbal

La comunicación no verbal (Pérez Feijoo et al., 2012) es el conjunto de formas de comunicación que no requieren del lenguaje verbal. Este tipo de comunicación es bastante importante ya que en una comunicación cara a cara supone el 65 % mientras que el componente verbal supone el 35 %, por lo que es la que mayor información transmite.

Este tipo de comunicación está relacionada con factores tales como los gestos, el movimiento que realizamos, los sonidos que producimos, la velocidad y el tono al hablar o la distancia personal. Pueden ser clasificados como factores asociados al lenguaje verbal y factores asociados al comportamiento (véase Figura 2.1). A continuación, se explicarán de forma más detallada:

- **Paralenguaje**: Factores asociados al lenguaje verbal. Comprende los sonidos no lingüísticos, es decir, aquellos sonidos que transmiten sensaciones o emociones, dependiendo de la velocidad del sonido y del volumen en el que se transmita.

- **Kinesia**: Factores asociados al comportamiento. Como tal, es una disciplina que estudia el significado de los movimientos humanos. Comprende la expresión facial, la mirada, la postura, los gestos y la háptica, que hace referencia al contacto físico.
- **Proxémica**: Factores asociados al comportamiento. Es el estudio de la relación espacial entre personas como manifestación social (véase Figura 2.2).

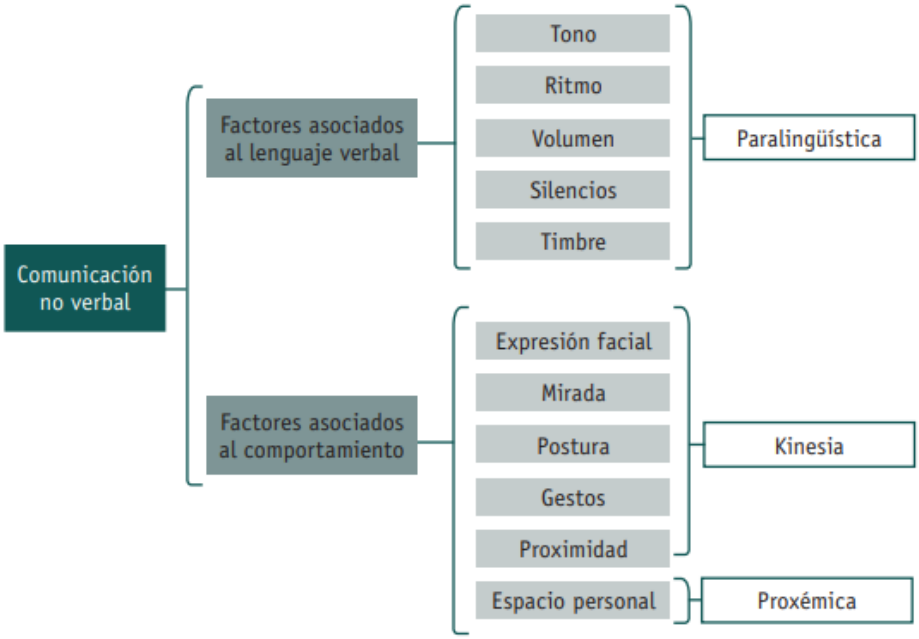



Figura 2.1: Factores asociados al lenguaje no verbal y al comportamiento (Pérez Feijoo et al., 2012)

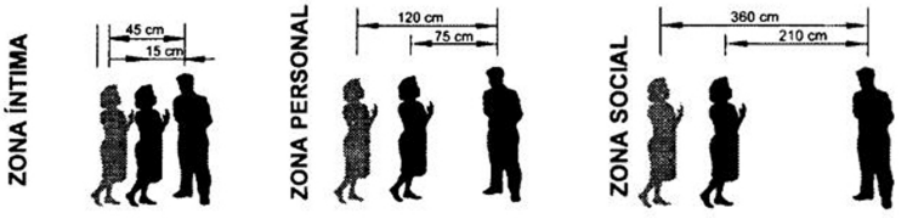



Figura 2.2: Tipos de distancia proxémica (Cano Moreno, 2015)

### 2.1.2. Rol del instructor

Una vez vistas las principales características de la comunicación no verbal como relación interpersonal, analizaremos qué es un instructor y qué requisitos se necesitan para ejercer dicho rol.

El instructor es aquel que interviene en el proceso de enseñanza y aprendizaje hacia otra persona, que tomaría el rol de aprendiz. Se encarga de facilitar el aprendizaje de una tarea en concreto, orientando y apoyando al aprendiz. Durante el proceso de la realización de la tarea en cuestión, el instructor irá evaluando al aprendiz y le irá dando las indicaciones pertinentes cuando sea necesario.

Existen diversos tipos de instructores, sin embargo este proyecto se centrará en los modelos de instructor tales como el instructor de autoescuela o el instructor personal deportivo, que se implican de manera más directa en la realización de una tarea.

Un instructor debe tener una serie de cualidades (Infoeducacion.es, 2018), que son indispensables para poder ejercer bien su rol. Las más importantes son las siguientes:

- **Paciencia**: cada persona afronta una situación de manera distinta. Por este motivo, el instructor deberá ser siempre paciente con su aprendiz y mostrar una actitud tranquila.
- **Empatía**: el instructor tiene que tener la capacidad de comprender los errores del aprendiz, ya que cualquier persona puede cometerlos.
- **Buena comunicación**: todas las explicaciones tienen que ser claras y concisas para que el aprendiz pueda realizar la tarea de la manera más eficaz.

## 2.2. Toma de decisiones en los videojuegos

La IA ha ido tomando cada vez más importancia en la industria de los videojuegos, ya que gracias a ella, los juegos van evolucionando y siendo cada vez más realistas. Actualmente un videojuego que tenga personajes o algún tipo de toma de decisiones táctica o estratégica va a estar usando al menos uno de los algoritmos más relevantes.

El uso de la IA nos permite poder crear personajes que tengan un comportamiento inteligente, ya sea a la hora de moverse, navegar por un espacio o muy especialmente a la hora de tomar decisiones.

Primeramente, se van a analizar los aspectos más importantes a considerar para decidir cuál es la mejor técnica de toma de decisiones (Duch i

Gavaldà y Tejedor Navarro, 2012). Uno de ellos es el tipo de situaciones que se pueden dar dependiendo de la información disponible. Existen dos tipos de situaciones:

- **Situaciones de certeza**. La IA tiene conocimiento de todos los posibles resultados en base a nuestras acciones, por lo que se puede calcular la situación más beneficiosa de todas las posibles y tomar las decisiones en base a ello.
- **Situaciones de incertidumbre**. La IA tiene conocimiento parcial, de tal forma que podemos estimar la situación resultante más beneficiosa en base a la información conocida.

Otro factor a considerar es en qué período de tiempo va a afectar la decisión tomada en el tiempo. Este factor se divide en tres tipos:

- **Decisión estratégica**. Este tipo decisiones tomadas llevan un tiempo en surtir efecto. Requieren de muchos datos de entrada, debido a que el éxito de esta decisión puede depender del resultado final.
- **Decisión operacional**. Este tipo de decisiones son inmediatas. No requieren de muchos datos, ya que pueden afectar al inicio de la partida.
- **Decisión táctica**. Este tipo de decisiones se encuentra a medias entre la decisión estratégica y la operacional. Se considera un subgrupo de elementos del juego y en base a ellos se toman las decisiones. Su función es optimizar la decisión estratégica.

El último factor a tener en cuenta es la frecuencia de la toma de decisión. Se divide en dos tipos:

- **Decisiones programadas**. Son aquellas que se toman de forma frecuente. De esta manera los métodos serán más sencillos y más concretos.
- **Decisiones no programadas**. Son aquellas que se realizan puntualmente. Los métodos en este caso serán más complejos y deberán estudiarse de forma más detallada, ya que son situaciones excepcionales.

### 2.2.1. El proceso de toma de decisiones

Una vez clasificados los distintos factores a considerar, se procederá a analizar el proceso de toma de decisiones (Duch i Gavaldà y Tejedor Navarro, 2012). Durante este proceso, existen unas etapas (véase Figura 2.3) necesarias para todos los algoritmos:


Figura 2.3: Proceso de toma de decisiones (Rodriguez y Rodriguez, 2011)

- **Identificar y analizar la situación**. Se recoge la información sobre la situación y se analizan las posibles acciones que se pueden realizar
- **Identificar los criterios de decisión y ponderarlos**. Una vez obtenida la información, se ponderan los datos según su relevancia.
- **Generar las alternativas de solución**. En esta etapa se decide qué decisión o decisiones se deben tomar para la situación actual en base a los datos ponderados.
- **Evaluar las alternativas**. Se analizan las decisiones elegidas para ver cuáles son más óptimas y se ponderan.
- **Elección de la mejor alternativa**. Se elige la mejor decisión, ya sea la más rápida o la más eficiente.
- **Implementación de la decisión**. En este apartado se ejecuta la decisión elegida.
- **Evaluación de los resultados**. Se valora el resultado obtenido al ejecutar esta decisión.

### 2.2.2. Técnicas de Inteligencia Artificial para la toma de decisiones

A continuación, se procederán a analizar los distintos tipos de algoritmos de IA que se usan para la toma de decisiones. Los algoritmos más importantes son los siguientes:

- **Sistema de reglas**. Este tipo de sistema funciona mediante la aplicación de reglas para evaluar estados y tomar decisiones. Es probablemente la herramienta más clásica que existe en el mundo de la IA y ha sido utilizada en algunos juegos, principalmente del género de estrategia. Sin embargo, resulta poco efectivo para situaciones poco comunes que no están debidamente recogidas dentro de las reglas creadas.
- **Máquinas de estados finitos**. Se trata de un modelo de comportamiento de un sistema con entradas y salidas las cuales dependen no solo de las señales de entrada actuales, sino también de las anteriores. Los estados que puede tener esta máquina son finitos y para que haya una transición de un estado a otro deben cumplirse unas condiciones específicas.
- **Árboles de decisión**. Se trata de un sistema encargado de analizar las diferentes opciones que podemos realizar. Tiene una estructura en forma de árbol, en la cual se colocan las decisiones posibles y sus consecuencias. Se empieza recorriendo desde la raíz hasta las hojas, hasta llegar a una decisión concreta.
- **Lógica difusa o probabilista**. Este tipo de sistemas permiten cuantificar ciertas observaciones que no se pueden clasificar como ciertas o falsas, sino que resultan relativas al observador. Establece términos intermedios asignando probabilidades a cada posible valor.
- **Redes bayesianas**. Es un tipo de sistema utilizado para representar la relación de dependencia condicional que existe entre un conjunto de variables aleatorias. Esto se representa a través de un grafo dirigido y acíclico, cuyos nodos corresponden a una variable aleatoria y están anotados con tablas de distribución.
- **Mapas de influencia** (Champandard, 2011). Es un algoritmo que se encarga de proporcionar información táctica sobre el mundo. Son útiles para procesos de decisión táctico-estratégica. Por ejemplo, pueden proveer información sobre las áreas controladas por enemigos, la frontera entre los territorios de aliados y enemigos, saber si un área ha sido atacada y recabar datos para predecir el movimiento del enemigo. Para generar este mapa se divide el mundo en distintas regiones y se

les asigna un valor. Este valor dependerá de lo que interese guardar como información relevante para la posterior toma de decisiones.

- **Árboles de comportamiento**. En inglés se conoce como *Behavior Trees* (BT) y es una técnica que supone la combinación de otras como las máquinas de estado y los árboles de decisión y que nos permite tanto optimizar el rendimiento como tener más variación en los comportamientos de la IA.

Nosotros nos enfocaremos más en los árboles de comportamiento, ya que es el algoritmo que más se adecúa al tipo de NPC que queremos crear.

### 2.2.3. Árboles de comportamiento

Estos árboles reemplazan las transiciones de las máquinas de estado con información de control adicional mediante unos nodos (nodos compuestos, que se explicarán más adelante), que deciden el orden en el que se tienen que ejecutar las tareas.

Los nodos que puede tener un árbol de comportamiento se suelen dividir en cuatro tipos: los nodos de acción, los nodos condicionales, los nodos compuestos, y los nodos decoradores.

Los nodos de acción (véase Figura 2.4) son aquellos que afectan directamente al juego, que representan la realización de tareas concretas como por ejemplo ir hacia un lugar, o interactuar con algún objeto.

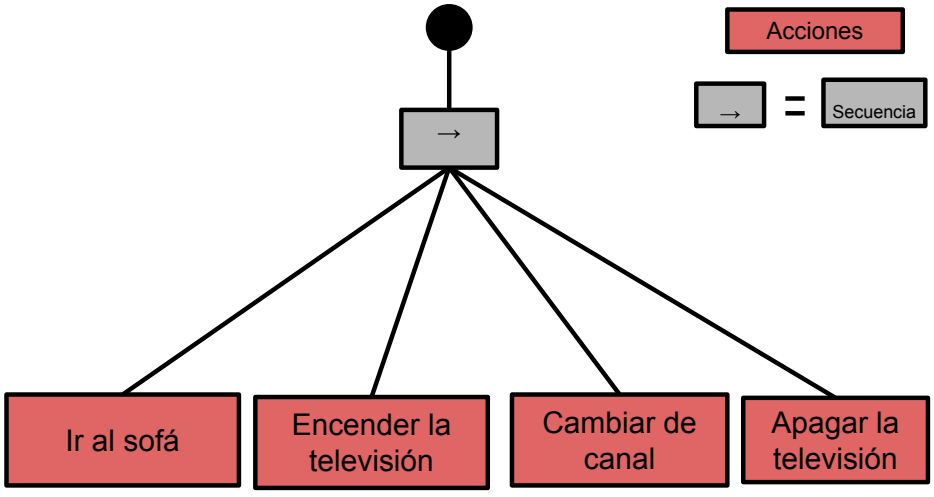



Figura 2.4: Ejemplo de árbol con nodos de acción

Los nodos condicionales (véase Figura 2.5) son aquellos se encargan de comprobar algún parámetro del juego, sin llegar a modificar el estado del mismo. En la siguiente imagen se pueden ver tareas condicionales como la de ver a un objeto o una persona.

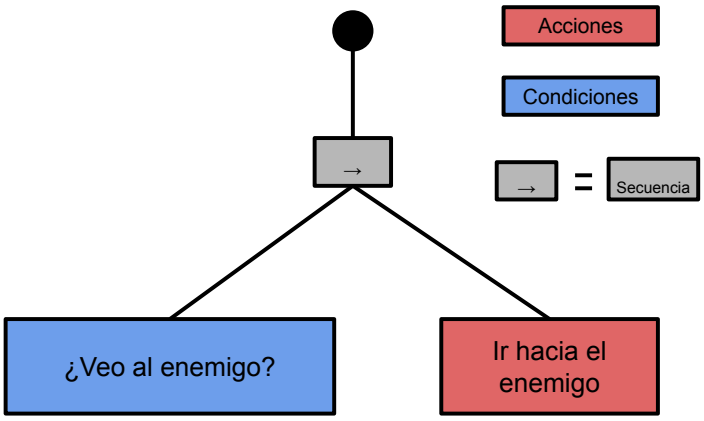



Figura 2.5: Ejemplo de árbol con un nodo de acción y un nodo condicional

Un nodo siempre va a devolver uno de los siguientes resultados:

- **Éxito** (*Success*). La tarea ha tenido éxito, por lo que se ejecuta.
- **Fallo** (*Failure*). La tarea no ha tenido éxito, por lo que no se ejecuta.
- **En ejecución** (*Running*). La condición para que termine la ejecución del nodo aún no se ha cumplido todavía, por lo que se sigue ejecutando la tarea.

Teniendo en cuenta estos estados, los nodos compuestos son aquellos que toman la decisión de cuáles son los siguientes nodos a ejecutarse.

Los nodos se van ejecutando de izquierda a derecha, por lo que se priorizan las distintas tareas según la posición donde estén dichos nodos.

En el caso de los nodos compuestos están formados por uno o más hijos. Su estado se basa en el resultado de evaluar a sus hijos en el orden habitual de izquierda a derecha.

Los nodos compuestos más relevantes son:

- ***Sequence***. Este nodo funciona como una puerta lógica *AND*, es decir, devolverá *Success* siempre y cuando todos los nodos hijos devuelvan *Success* y devolverá *Failure* si cualquiera de los hijos devuelve *Failure*.

- ***Selector*** (*Selector*). Este nodo, por otra parte, actúa como una puerta lógica *OR* (véase Figura 2.6), es decir, devolverá *Success* si alguno de los hijos devuelve *Success* y devolverá *Failure* si todos devuelven *Failure*.

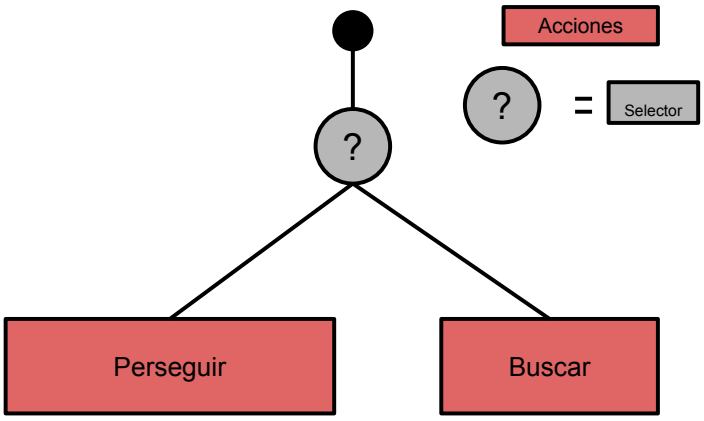



Figura 2.6: Ejemplo que utiliza el nodo *Selector*.

- ***Parallel***. Este nodo, menos frecuente, permite ejecutar simultáneamente los nodos hijos hasta que alguno devuelva *Failure* (véase Figura 2.7). Al igual que el nodo secuencia, devolverá *Success* si todos los hijos han devuelto *Success* y devolverá *Failure* si alguno de ellos devuelve *Failure*.

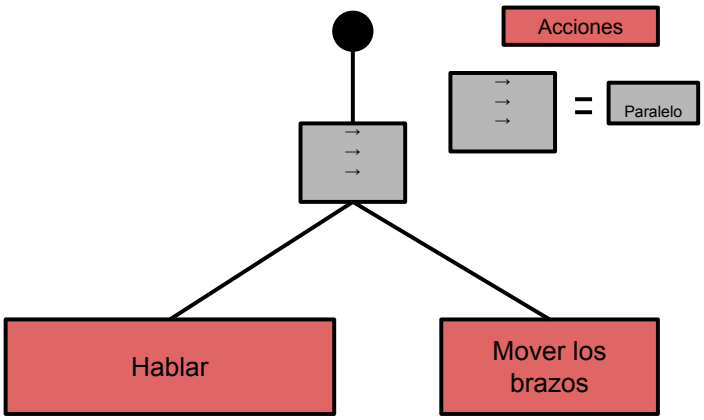



Figura 2.7: Ejemplo de nodo *Parallel*.

A diferencia de los nodos compuestos, los nodos decoradores solo pueden tener un hijo y pueden modificar el comportamiento o el estado del hijo. Los más comunes son los siguientes:

- ***Inverter*** (véase Figura 2.8). Funciona como una puerta *NOT*, es decir, si el estado que ha devuelto el hijo es *Success*, el inversor devolverá *Failure*, y viceversa.

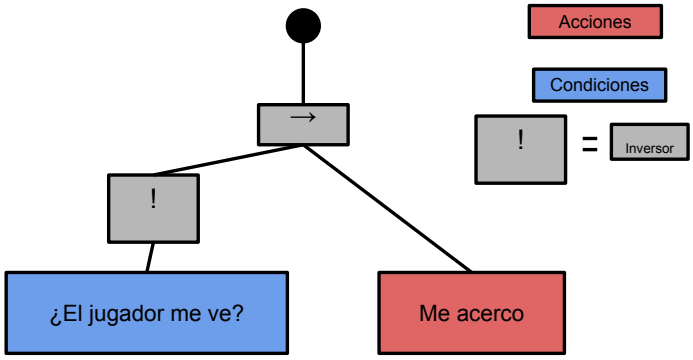



Figura 2.8: Ejemplo de nodo *Inverter*.

- ***Repeater*** (véase Figura 2.9). Modifica el número de veces que se ejecuta la tarea que tiene asignada como hija, ya sea un número finito o infinito de veces.

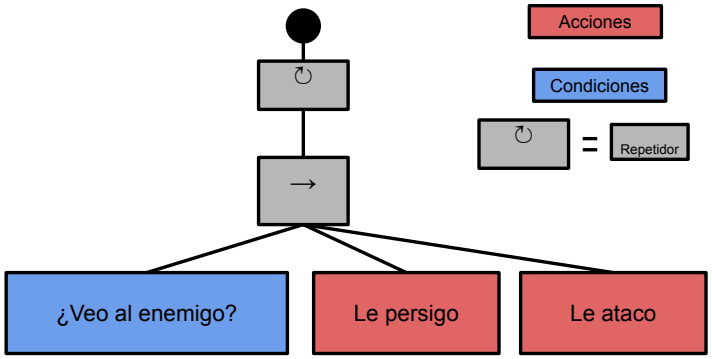



Figura 2.9: Ejemplo de nodo *Repeater*.

Otro aspecto importante de los árboles de comportamiento son los *conditional aborts* (Opsive, 2014c), también conocidos como *observer aborts*. Un **conditional abort** es una optimización que permite no tener que estar recomprobando el árbol entero en cada paso de ejecución. Esta optimización ha sido diseñada para que una tarea condicional pueda interrumpir a otras tareas y que estas dejen de ejecutarse en favor de otra que gana prioridad. Lo que ocurre en el proceso es que en ciertos momentos, aunque no corresponda por el orden prefijado en el árbol, se reevalúa la tarea condicional que tiene la capacidad de interrumpir y, si corresponde, la siguiente tarea se ejecuta-

rá según las reglas habituales de los árboles de comportamiento. Cualquier nodo compuesto puede tener aplicado un *conditional abort*.

Existen cuatro tipos de *conditional abort*:

- ***None***. Este es el comportamiento por defecto. En este caso la tarea condicional no se reevalua y no se ejecutará ninguna interrupción.
- ***Self***. En este tipo de comportamiento, la tarea condicional solo puede interrumpir a una tarea de acción si ambas tienen el mismo nodo compuesto padre. Esto se puede interpretar como que la tarea condicional se reevaluará en el momento en el que cualquier tarea dentro de la rama actual esté activa.
- ***Lower Priority***. Los árboles de comportamiento se puede organizar por prioridad. Si una tarea condicional con mayor prioridad cambia de estado, puede realizar una interrupción que parará la ejecución de las tareas menos prioritarias. Esto se puede traducir del tal forma que la tarea condicional será reevaluada en el caso de que cualquier tarea que se encuentre a la derecha de la rama actual esté activa
- ***Both***. Este comportamiento combina tanto el comportamiento Self como el Lower Priority. Puede ser interpretado de tal manera que la tarea condicional será reevaluada cuando cualquier tarea que se encuentre a la derecha de la rama actual o dentro de ella esté activa.

Como se puede observar, los árboles de comportamiento tienen algunas ventajas frente a otros algoritmos. Se pueden hacer cambios importantes de manera sencilla sin afectar demasiado al código, además de la posibilidad de crear árboles de comportamiento dentro de otros árboles, por lo que se mejora la creación de distintos módulos. Esto permite dividir mejor el trabajo y reducir el tiempo de desarrollo.

## 2.3. Experiencia de usuario en Realidad Virtual

Inmersión y presencia son dos conceptos diferentes, como explica en su tesis doctoral López Ibáñez (Ibañez, 2019). Durante este apartado explicaremos las características, relación y diferencias entre ambos conceptos.

La inmersión es un concepto objetivo sobre la semejanza de los dispositivos de RV con los sentidos y órganos humanos usados para percibir la realidad. No tiene la capacidad de ser medido.

Por otro lado, la presencia es un concepto medible, y se refiere a la capacidad de una experiencia en RV de provocar al usuario la sensación de "estar

ahí". Este concepto puede ser dividido en ideas más específicas, siendo las más comunes:

- **Telepresencia**: Mide como de fuerte es la sensación de una teletransportación física al escenario virtual, de una manera estrictamente sensorial.
- **Presencia social**: Mide la sensación de estar en un mundo virtual con otra persona. Solo se tienen en cuenta las interacciones sociales, no depende del escenario virtual.
- **Transporte mental**: Mide la calidad del transporte de conciencia de un usuario a un escenario virtual, y no tiene en cuenta la sensación física de teletransporte. Un usuario puede experimentar un transporte mental sin la sensación física de "estar ahí". Este puede ser el caso de muchos videojuegos con vista en primera persona.

La Psicología ha desarrollado formularios estándar para medir la presencia, los cuales utilizaremos para evaluar los prototipos en RV que realicemos. Los más habituales son Slather-Usoh-Steed (SUS) (Slater et al., 1994) y Temple Presence Inventory (TPI) (Lombard et al., 2009).

*Motion Sickness* (MS) y *Simulation Sickness* (SS) son dos conceptos diferentes que tienen similares efectos sobre los usuarios: mareos, nauseas y malestar. Su diferenciación viene dada por el origen de estos efectos. MS ocurre cuando hay movimiento físico en un entorno que aparentemente no está moviéndose desde la perspectiva del usuario, por ejemplo cuando este está sentado en un coche. Mientras que SS ocurre cuando no hay movimiento físico en un entorno que sí parece estar moviéndose, por ejemplo en un videojuego en RV con movimiento virtual.

SS es inversamente proporcional a la presencia, y debe ser reducido al máximo posible para obtener una experiencia de usuario placentera.

Los siguientes apartados van a tratar sobre los puntos más conflictivos de la RV para mantener la sensación de presencia en el videojuego.

### 2.3.1. Movimiento

El principal problema del movimiento es la fácil provocación de SS en el jugador. Esto es debido a que es muy común heredar de los videojuegos convencionales el uso del joystick del controlador de juegos para desplazarse, lo cual, si no se hacen ajustes en la aceleración y velocidad del movimiento ocasionado, puede provocar SS. Lo que obliga a los desarrolladores, si quieren mantener la sensación de presencia, a adaptarse a esta nueva plataforma y diseñar específicamente un método de movimiento para ella.

Un videojuego en RV que tenga movimiento continuo, especialmente a gran velocidad, suele provocar SS. Hay una gran cantidad de técnicas que permiten reducir la aparición de este SS, pero sólo es una reducción. Siendo una de las más utilizadas el tratar de que el movimiento del jugador sea siempre a velocidad constante, con el menor número de cambios en la aceleración posibles, reduciendo dinámicamente el campo de visión (FOV) en función del movimiento y la rotación, así como disminuyendo la complejidad de las texturas y evitando patrones estridentes en estas.

Oculus tiene una guía de buenas prácticas (Oculus), en la cual se explican los aspectos que se deben tener en cuenta a la hora de desarrollar un videojuego en RV y qué técnicas de locomoción son las más eficientes.

Una experiencia en RV pone a un usuario dentro de un mundo virtual. Este mundo tiene que mantener la coherencia con lo que la persona conoce respecto al mundo físico real. Es decir, si el jugador mueve la cabeza o se desplaza, este movimiento también ha de ser aplicado dentro del videojuego. Si queremos anular completamente el SS, debemos aplicar este principio a rajatabla. Esto obliga a los desarrolladores de RV a tener que buscar otras formas de permitir al jugador desplazarse o bien a diseñar todo su juego en torno a esta limitación.

Una de las formas más comunes de permitir el movimiento y evitar el SS es mediante el teletransporte, en el que desaparecemos y aparecemos casi instantáneamente en otra localización. Un ejemplo de está mecánica lo podemos encontrar en Budget Cuts (Neat Corporation, 2018) (véase Figura 2.10). Es una técnica muy útil para evitar el SS pero si se desea que el videojuego no tenga serias disonancias ludonarrativas (Hocking, 2007), tal vez no encaje en todo tipo de temáticas y ambientaciones narrativas.

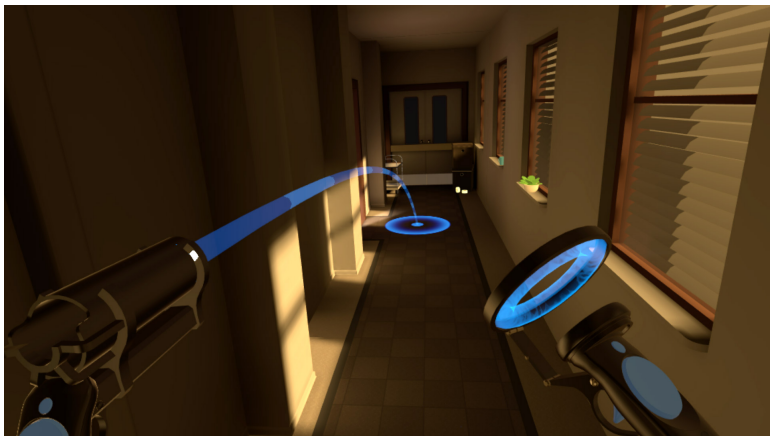


Figura 2.10: Teletransporte en el videojuego *Bugdet Cuts*

Otra de las alternativas es diseñar en torno a esta limitación, crear por ejemplo espacios cerrados en los que el movimiento esté restringido al área del jugador. Un ejemplo de juego exitoso aplicando este concepto es *Rick And Morty: Virtual Rick-Ality* (Owlchemy Labs, 2017) (véase Figura 2.11). La desventaja es que aceptar esta limitación implica no poder desarrollar todo tipo de videojuegos, pero con ella se puede obtener la mayor sensación de presencia posible y casi dejar de preocuparnos por el indeseable SS.


Figura 2.11: Ejemplo de movimiento restringido al área del jugador en el videojuego *Rick And Morty: Virtual Rick-Ality*

### 2.3.2. Interfaz de usuario

La interfaz de usuario es el punto de interacción entre el usuario y el videojuego. Su objetivo es brindar la información necesaria para que el usuario interactúe con fluidez durante el juego. Guiará al usuario de manera directa o intuitiva, para que pueda accionar y recorrer la narrativa correctamente (Rodríguez Christensen, 2018).

Una interfaz diegética es una interfaz que está incluida dentro del mundo del juego, es decir, que pueden ser vistas y escuchadas por los personajes. Las encontraremos como parte de la narrativa, integradas completamente en el universo del juego.

Existen dos problemas desde el punto de vista de la presencia al tratar con interfaces en RV:

- **Interfaces no diegéticas**. Una interfaz no diegética le esta dando información al usuario, al jugador, en vez de a su "personaje". Si bien no supone un problema de desconexión elevado con el mundo virtual, puede afectar levemente a la sensación de presencia.

- **Interfaces y SS**. Interfaces atadas al casco, es decir, que siguen el movimiento y rotación de la cabeza, tienen la capacidad de producir SS, especialmente las que se encuentran en el centro de la pantalla. Este tipo de interfaces no solo producen una elevada desconexión del sentimiento de presencia, sino que además producen malestar al jugador.

Como dijimos, Oculus dispone de una guía de buenas prácticas, y en ella existe un apartado que indica qué métodos no son recomendables a la hora de mostrar una interfaz al jugador y cual es la mejor solución para cada tipo de problema. Destacan evitar la oclusión de la escena con la interfaz y evitar dibujar elementos por detrás de algún objeto de la escena.

Las interfaces son un mal necesario en numerosas ocasiones. Se necesita dar información al jugador y es difícil encontrar la forma de que esta comunicación sea diegética. Pero si queremos la mayor sensación de presencia en nuestro videojuego, tenemos que tratar de limitarlas al máximo y encontrar otra manera de dar esa misma información al usuario.

También existe otra manera de dar información al jugador, y esto es mediante interfaces auditivas. Las interfaces auditivas también tienen el potencial de influir en el comportamiento de los jugadores, además de proporcionar una mayor sensación de presencia.

### 2.3.3. Interacción con el entorno

Los videojuegos en RV con 6 DoF (Mechatech, 2019) (del inglés *Degrees of Freedom*) y controladores en las manos son actualmente las experiencias para el público que tienen la capacidad de provocar la mayor sensación de presencia (véase Figura 2.12). Gracias a este hardware, tenemos la capacidad de movernos en un área cerrada con total libertad, nuestro posicionamiento con el cuerpo y rotación de la cabeza es traducido al videojuego. Además, los controladores se convierten en nuestras manos, siendo capaces de interactuar activamente con el entorno: tocar, agarrar, lanzar, golpear y usar herramientas.

Estas experiencias, otorgan al jugador una libertad de usuario que no disponen los videojuegos convencionales de mando o teclado. En estos últimos, las acciones que puede realizar el jugador son fácilmente limitables por el diseñador. Si este quiere que un escenario el jugador no sea capaz de interactuar con cierto objeto, lo único que tiene que hacer es no proporcionar esa interacción a ningún botón. Si suponemos el mismo escenario en RV, el jugador al ver un objeto en el suelo, puede agacharse y acercar su mano para cogerlo, si realiza la acción y no produce ningún resultado, buena parte de

Figura 2.12: Dispositivo RV con 6 DoF y controladores en la manos: *Oculus Quest*

la sensación de presencia se pierde y por tanto se ha roto la coherencia en el escenario.

Debido a la elevada libertad del usuario, mantener un nivel de presencia elevado requiere que el usuario sea capaz de interactuar con todo lo que tiene alrededor de su área jugable. Esto exige a los desarrolladores mantener una preocupación más elevada que en un videojuego convencional y construir una respuesta a todas las posibles interacciones. Esto es algo costoso de asumir, y que por limitaciones de presupuesto puede llevar a crear escenarios demasiado vacíos. Pero al menos lo que sabemos es que debe existir una diferenciación clara entre qué objetos son interactuables y cuáles no.

### 2.3.4. Personajes No Jugables

La RV permite tener mayor información acerca del estado del jugador. Las señales de entrada que se obtienen del usuario no están limitadas simplemente a activación de botones o la realización de movimientos de joystick o ratón, sino que se dispone de un flujo continuo de información, acerca de la posición y rotación de la cabeza y de las manos en cada momento del juego. Esta información da la posibilidad a los desarrolladores de detectar los gestos que hace el jugador, cómo reacciona ante eventos, su posición relativa en el escenario, etc. Usar toda esta información y proporcionar una respuesta adecuada hará que el jugador se sienta más presente en el mundo virtual.

Esta nueva tecnología de RV que eleva la sensación de presencia en el jugador y le hace estar mucho más inmerso en un mundo virtual, requiere de unos NPCs a la altura. El flujo continuo de entrada permite mejorar los comportamientos de estos.

Los NPCs en RV, al igual que los objetos interactuables, para mantener la sensación de presencia y debido a la libertad del usuario, deben proporcionar una respuesta adecuada a las acciones del jugador. Esto hace que su comportamiento sea más complejo que en videojuegos convencionales, ya que deben de estar atentos a todas las señales de entrada del jugador (incluso a las de inactividad) y proporcionar una respuesta coherente. Por ejemplo, si el jugador golpea a un NPC, este tiene que proporcionar alguna respuesta. Si no reacciona ante semejante interacción física, favorecerá que el jugador no se sienta presente en el escenario virtual.

## 2.4. Videojuegos de referencia

En esta sección se mostrarán los videojuegos que tomaremos como referencia para la realización del proyecto, centrándonos en escoger aquellos videojuegos en RV que hayan construido una buena interacción entre usuario y NPC. La lista de referencia mayoritariamente contiene videojuegos experimentales o demostraciones, pues actualmente en la gran industria no existen ejemplos de videojuegos que hayan construido una elaborada interacción con NPCs (véase Apéndice G).

### 2.4.1. *Shinobu Project*

*Shinobu Project* (Viva Dev, 2018) es un simulador de IA de la empresa centrado en la interacción del jugador con un NPC. En este juego se trata de pasar el día con un NPC, que será una chica que el jugador podrá personalizar a su gusto, y poder hacer tareas cotidianas con él. Se pueden hacer bastantes tareas como por ejemplo, cocinar, ayudar a asearse al NPC, hacerle fotos y varias cosas más.

El NPC tiene bastantes reacciones a las posibles acciones del jugador, tanto directas, como tocar la cara del NPC y que este se queje o darle la mano y que vaya contigo, como indirectas, que sería el caso en el que el NPC se duerme si el personaje apaga las velas de la habitación.

Las acciones más relevantes de este juego son las siguientes:

- Saludar al personaje (véase Figura 2.13).
- Coger la mano del personaje y poder llevarlo a cualquier parte.

Figura 2.13: NPC Saludando al personaje

- Dar objetos y recibir objetos de él.
- Hacer un gesto para que el personaje te siga.
- Golpear al personaje, provocando que se moleste
- Tanto el personaje como el jugador pueden cocinar.
- El jugador podrá jugar con el personaje a un juego llamado ***Chopsticks*** (Enacademic.com).
- Cuando el jugador llena la bañera de agua, el personaje antes de ducharse le echa del baño y no se ducha hasta que este se ha ido (véase Figura 2.14). Resulta ser una acción bastante compleja e interesante y que da mucho realismo al juego.

Figura 2.14: NPC esperando a que el jugador se vaya del baño

Las acciones, además de afectar de manera física al NPC, también afectan a sus emociones, provocando alegría o enfado dependiendo de cual sea la acción que se le hace al personaje (véase Figura 2.15).

Figura 2.15: NPC molesto por las acciones del jugador

Consideramos que es uno de los mejores ejemplos de juegos basados en la interacción con NPC, a pesar de encontrarse todavía en desarrollo.

### 2.4.2. *The Summer Camp*

*The Summer Camp* (Estudiofuture, 2017) (véase Figura 2.16) es otro ejemplo de videojuego orientado a la interacción entre el jugador y el NPC. Este juego utiliza una tecnología llamada VIVO creada por la propia empresa, especialmente desarrollada para proyectos en RV, que permite la creación de personajes realistas que reaccionan a nuestras acciones.


Figura 2.16: Imagen del videojuego *The Summer Camp*, que utiliza la tecnología *VIVO*

VIVO permite combinar animaciones con acciones físicas y comportamientos, ofreciendo a los desarrolladores nuevas herramientas para controlar

la actuación de los personajes dependiendo del estado del juego y la interacción del usuario. Facilita la creación de personajes que proporcionan reacción cuando les tocamos, cuando los miramos, cuando interactuamos con ellos, o incluso cuando simplemente no hacemos nada.

La herramienta consigue que no se repitan las mismas acciones y que no haya personajes parados que esperen a las acciones del jugador, dando realismo y fluidez al juego y proporcionando al jugador una mayor sensación de presencia.

*The Summer Camp* es una demostración de la herramienta. Esta consiste en interactuar con una serie de personajes, cada uno de ellos reaccionando de forma diferentes y tomando decisiones, afectando estas al desarrollo del argumento.

Las acciones más relevantes de este juego son las siguientes:

- Mirar al jugador cuando habla.
- Ofrecer objetos al jugador con la mano.
- Detectar que el jugador va a recoger un objeto y reaccionar antes de que lo haga.
- Mirar a lo más relevante de la escena.
- Ordenar acciones al jugador y reaccionar si no las realiza en el tiempo previsto.
- El jugador puede agarrar de la mano al personaje para ayudarlo a levantarse.
- Cuando el jugador toca el cuerpo del personaje reaccionan y se mueven.

### 2.4.3. Otros referentes

Cabe destacar también otras dos demos pertenecientes a Oculus que también utilizan la interacción del NPC con el jugador. Estas dos demos son: *Oculus First Contact* y *First Steps*.

En cuanto a *Oculus First Contact* (Oculus, 2016) (véase Figura 2.17), se trata de una demo donde podemos interactuar con un robot, que al principio se mostrará tímido con nosotros, pero más adelante se acercará y nos irá dando unas cintas para ir probando diferentes juegos o mecánicas. Podemos tener diferentes interacciones con el robot como por ejemplo saludarlo, que provoca que se acerque, coger objetos que nos da y si lanzamos cohetes, él seguirá su trayectoria con la mirada.

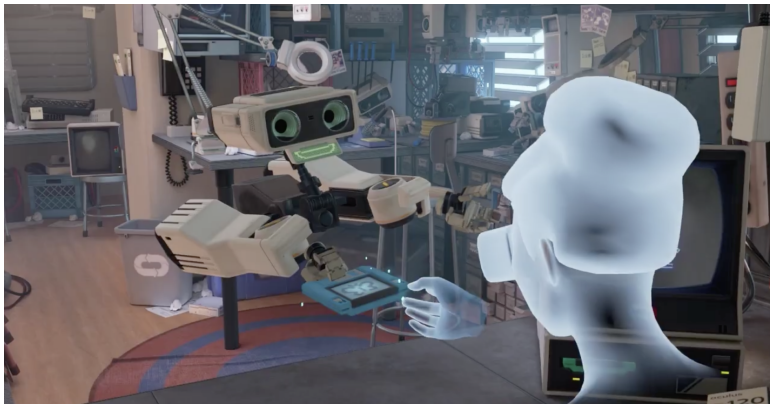


Figura 2.17: Ejemplo de interacción con un robot en *Oculus First Contact*

*First Steps* (Oculus, 2019) se trata de otra demo en la cual al igual que en Oculus First Steps, al meter diferentes cintas ocurren diferentes cosas, como por ejemplo que se active un juego de disparos o que aparezca un robot con el que podrás bailar. Cuando aparece este robot, al inicio se le podrá saludar y a continuación le podremos coger la mano para bailar con él (véase Figura 2.18).


Figura 2.18: Ejemplo de interacción con un robot en *Oculus First Steps*

Por último, un ejemplo reciente pero que se ha convertido en uno de los más relevantes en el mundo del RV es el videojuego *Half-Life: Alyx*. A pesar de que lo más destacable es la interacción con el entorno, que está muy lograda y es muy diversa, se puede apreciar alguna interacción con NPCs. Un ejemplo de esto es un escenario en el cual el jugador se encuentra dentro de un ascensor (véase Figura 2.19) y al abrirse las puertas aparecen unos enemigos apuntándole con armas. Si el jugador trata de pulsar continuadamente el botón del ascensor para que se cierren las puertas, el enemigo más cercano disparará hacia el botón y le dirá al jugador que pare de tocarlo. En ese

momento, además, si el jugador le intenta quitar el arma al enemigo, este se echará hacia atrás.

Figura 2.19: Ejemplo de interacción con un NPC en *Half-Life: Alyx*

Capítulo 3

# Objetivos y especificación

El propósito general de este proyecto es desarrollar una herramienta, en forma de complemento para Unity, que permita construir NPCs que mejoren la presencia social en videojuegos de RV.

Un NPC verdaderamente interactivo debe realizar una constante monitorización del jugador para reaccionar a todas las acciones relevantes de su avatar. Hay muchas acciones básicas a las que todo personaje debería dar respuesta, como una fuerte colisión, por ejemplo. Sin embargo otras dependen del contexto, y por ello es necesario permitir la creación de acciones personalizadas así como nuevas condiciones que permitan activar estas acciones.

En este capítulo se detallan los objetivos del trabajo y se especifican todos los requisitos que debe cumplir el sistema informático a desarrollar.

## 3.1. Objetivos

Los objetivos principales del proyecto son los siguientes:

- Identificar los factores que tienen mayor influencia en la sensación de presencia que tiene un jugador en RV.
- Desarrollo de una herramienta en Unity que permita crear NPCs cuyo comportamiento suscite presencia social y que sea altamente configurable.
- Creación de un escenario de ejemplo que contenga un NPC, cuyo rol sea el de instructor, utilizando la herramienta desarrollada.

- Comprobar la eficacia de la herramienta realizando pruebas con jugadores reales y midiendo la repercusión en la sensación de presencia.

## 3.2. Especificación

En las próximas subsecciones se especifican las limitaciones y los diferentes requisitos de la herramienta a desarrollar.

### 3.2.1. Limitaciones

El desarrollo de esta herramienta requiere establecer limitaciones previas, de lo contrario, no alcanzaríamos los objetivos por falta de tiempo y recursos. Además, ayuda a clarificar la especificación de los requisitos, evitando abrir un abanico excesivamente amplio de posibilidades.

Hemos establecido las siguientes limitaciones:

- **Comunicación no-verbal con el NPC**. Debido a que queremos que la inmersión, credibilidad y sensación de presencia sea lo más elevada posible hemos establecido que la comunicación se base en lenguaje no verbal. Si la comunicación fuese verbal, existirían dos posibilidades:

  1. **Comunicación oral**. Si el NPC se comunicase oralmente, y lo hace en un idioma que entendemos, lo coherente para mantener la inmersión sería que nosotros también pudiéramos hablar, lo que no es viable con la tecnología de que disponemos.
  2. **Comunicación escrita**. Consideramos que, a parte de la complejidad, una comunicación escrita sería muy pausada y limitaría la sensación de presencia, por lo que decidimos descartarla.

  Por lo tanto, concluimos en utilizar comunicación no verbal, dando peso a los gestos y posición del usuario y del NPC, así como a reacciones paralingüísticas del NPC.

- **Animaciones no orgánicas**. El NPC que crearemos para la demostración de la herramienta no tendrá un modelo de animación humano. Solo dispondrá de los elementos básicos para poder reaccionar al usuario, y hemos considerado que esos elementos son cabeza, ojos, tronco superior y brazos. Esta limitación no implica que la herramienta no pueda utilizarse con NPCs que utilizan animaciones humanas, es simplemente una restricción para la demostración de la herramienta.

- **Detección de gestos**. Nuestro proyecto no pretende profundizar en el complejo campo de la detección de gestos y por lo tanto la herramienta

recurrirá a recursos y bibliotecas externas para reconocer gestos, que además no serán muy sofisticados.

### 3.2.2. Tareas y comportamientos del NPC

Para la construcción de la herramienta se ha hecho una clasificación de los diferentes comportamientos del NPC. Este desempeñará **tareas**, cuya división principal estará formada por **acciones** y **condiciones**.

A continuación especificaremos las **tareas** que hemos considerado básicas para lograr que un NPC genérico resulte lo más realista posible y que, por tanto, la herramienta tratará de implementar.

Además, se especificarán los **conjuntos de comportamientos** más comunes en todo NPC interactivo. Un conjunto de comportamientos es una sucesión de acciones y/o condiciones.

#### Acciones

Las **acciones** más básicas consideradas son las siguientes:

- **Estar quieto**. Habrá ocasiones en las que el NPC se tenga que quedar quieto porque no tenga nada que hacer en ese momento. Se quedará mirando en la dirección donde esté en ese instante, aunque el jugador se mueva hacia otro lado.
- **Esperar**. El NPC antes o después de haber realizado una acción, puede esperar unos segundos antes de continuar con lo que tenga que hacer.
- **Acciones corporales**. Para mostrar la reacción ante las distintas acciones del jugador, el NPC se expresará haciendo gestos con el cuerpo, ya sea con las manos o con la cabeza.
- **Comunicación sonora**. El NPC puede comunicarse con el jugador a través de secuencias de sonidos.
- **Efectos especiales**. Es necesaria la utilización de efectos especiales para mostrar reacciones ante ciertas acciones del jugador. Por ejemplo, un efecto de humo cuando el jugador quema al NPC.
- **Movimiento**. El NPC normalmente estará atento al avatar del jugador y a sus acciones, por lo que mantendrá una distancia prudente con él. En caso de que este se aleje demasiado, el NPC le seguirá, pero no será el único movimiento que realice. Por ejemplo, habrá ocasiones en las que el NPC podría necesitar acercarse a una zona concreta o a un objeto para ayudar en la explicación de la tarea que tiene que realizar

el jugador, o bien huirá de la posición en la que se encuentra si el jugador le está incomodando.

- **Rotación**. Existen diferentes acciones de rotación, como por ejemplo girarse hacia el jugador para demostrar que le está atendiendo, o darle la espalda como señal de estar molesto con él.
- **Interacción con objetos**. Al igual que cualquier ser humano, el NPC puede coger objetos que estén repartidos por el escenario y si para realizar una tarea concreta el jugador tuviera que coger algún objeto, el NPC podría ayudarle y acercárselo él mismo.
- **Enseñar cartel**. Dado que es habitual no contar con comunicación verbal, como dijimos antes, el NPC puede mostrar carteles para poder transmitir alguna información o algunas explicaciones al jugador.

### Condiciones

En cuanto a las **condiciones**, podemos clasificarlas en las siguientes categorías:

- **Condiciones corporales**. Este tipo de condiciones pertenecen a los campos de la Kinesia y la Proxémica, exceptuando los gestos que se hacen con las manos. Las consideradas más relevantes son las siguientes:
  - **Ver al jugador**. A menudo el NPC necesita ver al avatar del jugador, para estar al tanto de lo que está haciendo y poder ayudarle en caso de que lo necesite.
  - **Ser visto por el jugador**. El NPC necesita saber si el avatar del jugador le está mirando, es decir, si este le está prestando atención. Si alguien se queda mirando al NPC durante mucho tiempo puede significar que este jugador necesita ayuda y posiblemente desea que el NPC le ayude.
  - **El jugador da la espalda al NPC**. Comprueba si el jugador está dando la espalda al NPC, dando a entender que el jugador no está prestando atención al mismo.
  - **Jugador quieto**. El NPC reaccionará en caso de que el jugador se mantenga quieto durante mucho tiempo, porque puede interpretar que el jugador se encuentra perdido.
  - **El jugador está agachado**. Que una persona esté agachada suele implicar que algo se le ha caído o que ha perdido algo y lo está buscando, por lo que, de nuevo, el NPC podría ayudarle.

- **Interacción del jugador con el entorno**. Condiciones que detectan cuando el jugador interactúa algún objeto que se encuentre por el escenario. Los más relevantes son los siguientes:
  - **El jugador coge un objeto**. Dependiendo de qué objeto sea, el hecho de que el jugador coja un objeto puede ser relevante para el NPC, por ejemplo si se tratara de un objeto erróneo.
  - **El jugador usa un objeto**. El NPC podría reaccionar si el jugador usara por ejemplo un objeto peligroso.
  - **El jugador señala a un objeto**. Que el jugador señale a un objeto con alguna de las manos puede dar a entender al NPC que siente interés en ese objeto en ese momento y darle alguna respuesta ante ello.
  - **El jugador se teletransporta**. Ya que el NPC está cerca del jugador en todo momento para estar pendiente de él, si este se teletransporta, el NPC le seguirá hasta la zona a la que se haya teletransportado.
- **Interacción física del jugador con el NPC**. Conjunto de condiciones que se basan en el contacto entre el NPC y el jugador, siendo las más relevantes:
  - **Ser tocado por el jugador**. Si el jugador toca el NPC este se sentirá incómodo y huirá, ya que en la mayoría de los casos no se espera un contacto tan íntimo entre ambos.
  - **Ser golpeado por un objeto**. Si el jugador le tirá un objeto al NPC esté se mostrará molesto con el jugador.
  - **Ser golpeado por el jugador**. Si el jugador golpea al NPC, este se enfadará con el jugador.
  - **Otros**. Posibles condiciones más complejas y menos relevantes pero que resultarían útiles para crear mejores comportamientos entre ambos. Entre estos ejemplos podrían encontrarse recibir algo por parte del jugador, chocar las cinco, ser empujado, saludar al jugador dándole la mano y otras muchas más.
- **Gesticulación del jugador**. En este apartado se contemplan las condiciones de gestos que pertenecen a la Kinesia, que requieren de las manos o la cabeza para interactuar y no requieren de contacto físico.
  - **Gestos con la cabeza**. El NPC puede reaccionar ante posibles gestos que realice el jugador con su avatar como asentir o negar con la cabeza. Gestos que, por ejemplo, pueden transmitir información al NPC sobre si el jugador ha entendido o no una explicación.

- **Gestos con las manos**. Diversas condiciones que requieren de gesticulación con las manos con diferentes significados.
  - **Saludar**. Gesto básico en la interacción humana para iniciar la comunicación. Si el jugador saluda al NPC, el NPC le devolverá el saludo de vuelta.
  - **Señalar**. Si el jugador señala al NPC, puede dar a entender que espera algo de él.
  - **Levantar la mano**. Si el jugador levanta la mano, el NPC interpretará que tiene dudas sobre lo que tiene que hacer.
  - **Otros**. Otros posibles gestos no básicos en la comunicación pero que aportarían una mayor complejidad y riqueza a esta serían: aplaudir, taparse la cara, cruzarse de brazos, levantar los dos brazos, extender las dos palmas de las manos hacia delante, un gesto de *OK* y muchos otros más.

### Conjunto de comportamientos

Tal y como se ha mencionado anteriormente, un **conjunto de comportamientos** es una sucesión de **acciones** y **condiciones**. A continuación se especifican aquellos que pueden ser dados a la mayoría de NPCs interactivos:

- **Quedarse quieto y mantener al jugador a la vista**. Habrá ocasiones en las que el NPC se tenga que quedar quieto porque no tenga que dar ninguna indicación como tal, ni acercarse hacia él porque no se está moviendo de su zona en ese momento. A pesar de estar quieto, mantiene el contacto visual con el NPC y se gira hacia la dirección en la que esté.
- **Mantener el contacto visual con el jugador**. Como es normal en una relación entre dos personas, en la mayoría de situaciones, el NPC tratará de mantener el contacto visual con el NPC, y estará pendiente de él en caso de que tuviera alguna duda en algún momento.
- **Ayudar al jugador con alguna tarea**. Para poder agilizar la tarea que tiene que realizar el jugador, en algunos momentos el NPC le ayudará como por ejemplo llevándole algún objeto que sea necesario.
- **Comunicación**. El NPC a pesar de no ser humano, tiene su propio idioma, que es una secuencia de sonidos u onomatopeyas.
- **Dar explicaciones**. El NPC tendrá que explicar al jugador qué es lo que tiene que hacer en cada momento mediante carteles con símbolos, mientras se comunica y mueve los brazos, que es un gesto común a la hora de realizar una explicación.

- **Seguir al jugador si se aleja**. Ya que el NPC es un instructor, deberá estar atento a las acciones y movimientos que vaya realizando el jugador. Para ello, deberá mantener una distancia prudencial a la que le pueda observar y acercarse a él en caso de que se vaya a otra zona de la habitación.
- **Reaccionar cuando tenga algún problema con el jugador**. A pesar de que el NPC no sea humano, también siente y piensa de manera similar a uno. Es por esto, que si en algún momento el jugador tratara de agredirle, ya sea mediante un golpe o porque le ha lanzado algún objeto a la cabeza, este reaccionará o bien huyendo o bien mostrándole su enfado.
- **Saludar como primer gesto a realizar**. Ya que el saludo es una interacción básica al iniciar una comunicación, será lo primero que el NPC haga antes de empezar la prueba, ya sea porque él ha decidido saludar primero, o como respuesta al saludo del jugador.
- **Comportamiento proxémico**. Como ya se ha explicado en el estado de la cuestión, la Proxémica es importante en una relación entre dos personas. El NPC tratará siempre de mantener una distancia social con el jugador, ya que a menudo son dos personajes que no se conocen o aunque se conozcan no comparten una excesiva intimidad. Si el jugador sobrepasara la distancia social, para entrar en una distancia excesivamente personal entre él y el NPC, este se sentirá incómodo y se lo hará saber, ya que está invadiendo su espacio privado. Si el jugador quiere "dar un paso más" y se acerca aún más al NPC, invadiendo su intimidad, este huirá hacia atrás, molesto. El NPC también se sentirá molesto en caso de que el jugador le toque, ya que asumimos que salvo casos especiales, no hay confianza suficiente entre ambos personajes para una acción así.

### 3.2.3. Demostración de la herramienta

Habiendo completado el desarrollo de la herramienta, se creará una demostración de la misma creando una escena de ejemplo. Esta demostración consistirá en una escenario interactivo en la cuál habrá un NPC cuyo rol es el de **instructor**. El jugador tendrá un objetivo y este NPC irá proponiendo una serie de tareas al jugador, que tendrá que ir realizando mediante la interacción y el traslado de ciertos objetos que habrá repartidos por el escenario.

Esta demostración de la herramienta tiene **dos objetivos:**

1. Servir como **ejemplo** de uso de la herramienta para los desarrolladores, mostrando cómo se construyen las **tareas** (acciones y condiciones) y los **comportamientos** propios de un videojuego concreto.

2. Conseguir una escena con un NPC que proporcione mayor sensación de presencia social que uno construido sin la herramienta, pudiendo así validar esto mediante experimentos de tipo **test A/B** con usuarios reales.

El funcionamiento de la demostración será exactamente el mismo en ambas escenas A y B, excepto por el comportamiento del NPC. En la escena A, actuará como un NPC convencional de cualquier otro videojuego. No reaccionará a las acciones del usuario y solo se encargará de dar una explicación inicial y seguirte a donde vayas.

Por otro lado, el NPC de la escena B utilizará todas las tareas mencionadas en el apartado anterior para conseguir un comportamiento interactivo, al que se le añadirán tareas y comportamientos propios del videojuego en cuestión.

Se intentará facilitar la **extensibilidad** de la herramienta, puesto que cada videojuego tiene sus características particulares y por este motivo, la herramienta no puede dar respuesta a todas las posibles combinaciones, y por tanto, debe ser posible adaptar los NPCs a cada juego.

Por ejemplo, la herramienta facilitará detectar cuando un jugador levanta la mano, sin embargo, si el videojuego a desarrollar necesita la detección de un gesto de asustarse, tendrá que ser implementado por el propio programador. La herramienta proporciona únicamente las facilidades para los comportamientos más genéricos y aquellos más específicos que han sido necesarios para realizar la escena de ejemplo.

### 3.2.4. Interfaz

Toda la herramienta se basa en la utilización de una herramienta que permite implementar **árboles de comportamiento**, como es ***Behavior Designer*** (Opsive, 2014a). Por lo tanto, su programación será similar y en cierta medida dependiente de este. Los recursos que representan árboles de comportamiento se basan en la creación de **tareas** (acciones y condiciones) y **conjuntos de comportamientos** como ya se ha comentado antes.

El proyecto será empaquetado como un conjunto de tareas y comportamientos referentes a la presencia social de NPCs en RV, para simplificar su integración dentro de los árboles de comportamiento. Esto funcionará de manera similar a cómo lo hacen otros paquetes con tareas de diferente índole,

como por ejemplo Behavior Designer Formations Pack (Opsive, 2015) para *Behavior Designer*.

El árbol de comportamiento asociado a un NPC es el punto de partida de nuestra herramienta. Su funcionamiento es ilustrado mediante la propia interfaz de *Behavior Designer*. El funcionamiento de los diferentes paquetes de árboles de comportamiento es muy similar, no existen diferencias significativas a la de hora de interactuar con ellos, por lo que el manejo de la interfaz de nuestra herramienta será básicamente el de *Behavior Designer*.

Como se puede observar en la Figura 3.1, dentro del menú de creación de una tarea, se puede especificar si es una acción o condición, y dentro de esta categoría utilizar alguno de los nodos de *Behavior Designer Formations* que se encuentran en **Formations**.

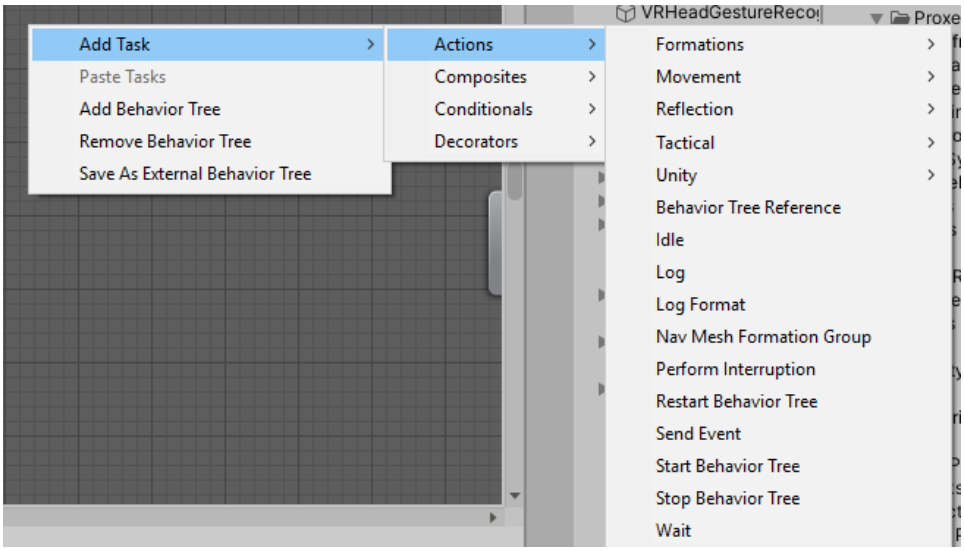



Figura 3.1: Ejemplo de creación de tarea en *Behavior Designer*

Nuestra herramienta aparecerá dentro de este mismo menú, pudiéndose encontrar todas las tareas desarrolladas dentro de una nueva categoría: **SocialPresenceVR**.

Como ya hemos mencionado, en la herramienta habrá fundamentalmente dos tipos de tareas, las acciones y las condiciones, cada una con su respectivo icono para poder distinguirse de manera clara.

Cada tarea tendrá sus propios parámetros, que podrán ser personalizados en el menú del inspector del *Behavior Designer* (véase Figura 3.2), en lugar de tener que acceder al código y tener modificar las variables desde ahí. Cada uno de los parámetros puede ser configurado mediante un valor o una variable.

Si se coloca el cursor encima de cada variable, aparecerá una breve explicación que indica su significado. Esta opción es muy útil para variables cuyo nombre pueda no ser autoexplicativo.

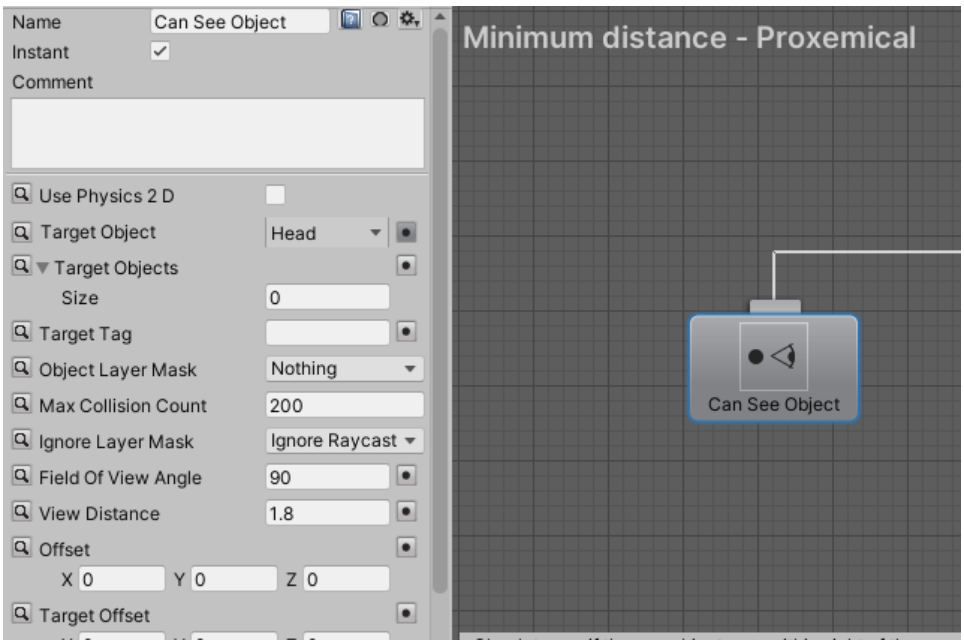



Figura 3.2: Ejemplo de personalización de tarea en *Behavior Designer*

Capítulo 4

# Metodología

En este capítulo se hablará del modelo de proceso y planificación general llevado a cabo para la realización del proyecto, las diferentes iteraciones e ideas que han ido surgiendo a lo largo de este y por último. se mencionarán las herramientas utilizadas para poder realizar la metodología planteada.

## 4.1. Modelo de proceso y planificación general

Para poder llevar a cabo la producción del proyecto, hemos hecho uso de las metodologías ágiles, que hemos aprendido en la asignatura de *Metodologías ágiles de producción* y aplicado durante la carrera en las asignaturas de *Proyecto 1*, *Proyecto 2* y *Proyecto 3*.

Las metodologías ágiles son procesos que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

Entre sus ventajas se encuentran algunas como la mejora de la satisfacción del cliente, dado que estará involucrado y comprometido a lo largo del proyecto, la mejora en la motivación e implicación del equipo de desarrollo, el ahorro en costes y tiempo, y una mayor velocidad y eficiencia.

Por limitaciones naturales del curso académico, hemos estructurado la realización del proyecto en tres grandes etapas, llamadas hitos, las cuales han comprendido los periodos: desde comienzo de curso a Navidad, desde Navidad a Semana Santa, y desde Semana Santa hasta hasta el Hito Final (que incluye la revisión y la posterior defensa del Trabajo Fin de Grado). Cada uno de estos periodos tuvo una duración aproximada de tres meses.

Dentro de cada hito, se han ido elaborando las tareas necesarias para cada *sprint* y se han ido teniendo reuniones con el tutor para supervisar el trabajo realizado hasta el momento. Los *sprints* son periodos cortos de tiempo, en nuestro caso han sido de dos semanas, y por lo tanto, las reuniones acordadas con el tutor se han realizado generalmente en el mismo intervalo.

En un inicio, se estimó un esfuerzo total de 360 horas de trabajo por cada uno de los integrantes para la realización completa del proyecto. Los objetivos que se han ido marcando se han ajustado en base a las horas disponibles en cada hito.

- **Hito de Navidad**. El objetivo de este hito se ha basado fundamentalmente en el desarrollo de la idea, que ha ido variando bastante a lo largo del tiempo hasta que se ha consolidado finalmente. Se ha investigado sobre qué herramientas utilizar para el desarrollo y se ha comenzado un prototipo para probar el correcto funcionamiento de las herramientas elegidas. Durante este hito se han realizado una media de 14h (7h cada uno) por sprint.
- **Hito de Semana Santa**. Este hito ha estado orientado especialmente al desarrollo de la herramienta y el asentamiento de las herramientas a utilizar. Durante este hito se han realizado una media de 47h (23h cada uno) por sprint.
- **Hito Final**. Este hito ha servido para poder arreglar errores, optimizar tareas de la herramienta y asegurar el correcto funcionamiento de el escenario de ejemplo. Se han creado también los cuestionarios de evaluación, de los tests A/B, que han sido entregados a distintos usuarios, para poder comprobar la utilidad del proyecto. Durante este hito se han realizado una media de 76h (38h cada uno) por sprint.

Cada tarea que hemos ido completando tiene un valor que equivale al número de horas que han sido necesarias para llevar a cabo dicha tarea. Se ha establecido que un punto de una tarea equivale a una hora empleada. Todas las tareas desarrolladas en este proyecto pueden ser visualizadas en nuestro tablero de Trello[1]. La planificación estimó un total de 360 horas por integrante, sin embargo, el desarrollo ha acabado con 450 horas por integrante.

## 4.2. Iteraciones y generación de ideas

A lo largo de la etapa inicial de diseño de la idea, ha habido diferentes procesos de generación de ideas, que han dado lugar a varios enfoques y

[1] `https://trello.com/b/ZKIyIzPx/tfg-castillo-serrano`

modificaciones hasta dar con la propuesta definitiva, que es la mencionada y explicada en la memoria.

### Primera propuesta

La idea inicial estaba basada en la realización de un juego donde fuese relevante la implicación del jugador en el entorno donde se encontraba. En ella, se planteaba un escenario de tensión como un atraco o un robo en el cual el jugador tuviera una gran relevancia y tuviera que tomar una serie de decisiones complicadas que afectaran o bien al entorno o bien a otros personajes.

Este planteamiento no duró demasiado tiempo, puesto que en seguida se abordó una segunda opción que resultaría más interesante de estudiar y analizar.

### Segunda propuesta

Esta segunda propuesta se basó en la realización de un proyecto orientado a la respuesta del jugador a la autoridad (como por ejemplo un personaje instructor). Dicho proyecto se apoyaba en los experimentos de Milgram (Milgram y Gudehus, 1978) y de la cárcel de Stanford (Zimbardo, 1972).

La idea consistía en la realización de una serie de pruebas en las cuales el jugador se vería obligado a hacer acciones que irían siendo cada vez menos morales y más violentas. Se trataba de estudiar hasta dónde es capaz de llegar un ser humano ante unas órdenes recibidas por un NPC dentro un entorno de RV.

Esta idea acabó siendo descartada, debido a que se fundamentaba demasiado en aspectos referentes al mundo de la Psicología, campo en el cual no estamos especializados, y por el cual, necesitaríamos ayuda y asesoramiento de terceros para poder llevar dicha idea a cabo.

Es por esto, que se volvió a redefinir una vez más el concepto, para que este fuera más claramente orientado al mundo de los videojuegos y a la utilidad para su industria, en lugar de estar orientada a la observación de resultados psicológicos.

Esta situación dio paso al planteamiento de la tercera y última propuesta, que resultaría ser más interesante como Trabajo de Fin de Grado, en la cual podríamos aprender más y que podría ayudar más a los desarrolladores de videojuegos para RV.

### Tercera y última propuesta

Para poder asentar la idea final, se procedió a la búsqueda y análisis de diferentes videojuegos de RV y se realizó una clasificación para estudiar sus características comunes (véase Figura 4.1). La tabla completa puede encontrarse en el Apéndice G.

| JUEGOS | TIPO DE MOVIMIENTO | AVATAR | INTERFAZ | INTERACCIÓN CON ENTORNO |
|---|---|---|---|---|
| No Man's Sky | Continuo - Joystick*, Salto - botón* | Manos | Estática en la escena, Interfaz atada al casco | Interacción realista con palancas y volante |
| Defector | Continuo - Joystick* | Todo el cuerpo aunque invisible | Interfaz atada al casco | Interacción realista con botones y escalar |
| SuperHot | Presencial - No joystick. | Manos | Interfaz en la escena | Interacción realista con objetos, se lanzan |
| Vader Immortal | Presencial - No joystick + teleport no fijo | Manos con mando | Interfaz en la escena | Interacción realista con botones y palanca. |
| Space Pirate Trainer | Presencial - No joystick | Sin manos- Únicamente arma | Interfaz en la escena | Interacción realista con armas |
| Fallout VR | Presencial + Joystick + teleport no fijo (Te puedes mover a cualquier mueble o cosa, no solo al suelo) | Se ven los mandos si no coges armas | Interfaz atada al casco. Manejo de menú con el PAD | Interacción no realista. Se hace con el PAD. |

Figura 4.1: Pequeña muestra del análisis de diferentes juegos de RV

Se centró en estudiar los siguientes aspectos:

- **Tipo de movimiento**. Si se implementaba el movimiento mediante un mando, movimiento presencial del jugador o bien mediante teletransportes.
- **Avatar**. Si el avatar funciona mediante un controlador de juego o mediante las manos.
- **Interfaz**. Si la interfaz existente se encuentra en la escena o asociada al visor de RV.
- **Interacción con el entorno**. Si la interacción con el entorno es mediante las manos o mediante utilización de botones.
- **Tutorial**. Si hay o no tutorial en el juego.
- **Extras**. Si hay alguna característica especial a destacar del juego, como por ejemplo si se trata de un *port* directo de un juego para PC o consola.
- **Menús**. Si hay o no menús y si los hay, si son 3D o 2D.
- **Interacción con los objetos**. Si se pueden coger objetos o no, o si a la hora de poder coger un objeto, se indica resaltándolo.
- **NPC**. Si hay o no NPC, si son o no realistas e interactivos.

Una vez realizado el estudio, se tuvieron en consideración dos aspectos:

- Por una parte se tomaron como base las características comunes y mejor orientadas a un juego en RV para poder implementarlas en nuestra escena de ejemplo y que fuera lo más realista posible.
- Por otra parte, se destacó especialmente la columna de los NPCs, en la cual la mayoría de ejemplos de videojuegos o carecen de NPC o los NPC que existen no son realistas ni suficientemente interactivos. (véase Figura 4.2)

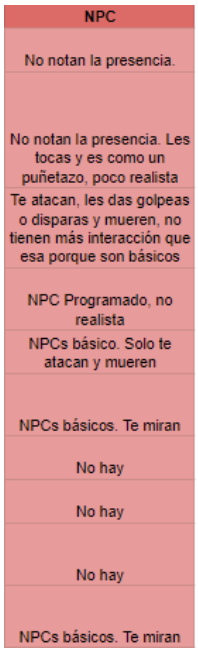



Figura 4.2: Pequeña muestra del estudio de los NPC

Resultó bastante relevante que, en general, los desarrolladores no se enfoquen en crear NPCs interactivos y realistas, aspecto que resulta innegablemente importante para la experiencia del jugador.

Se han visto varios ejemplos de gameplays de videojuegos en los cuales el jugador tocaba al NPC o le tiraba algún objeto y este no reaccionaba en absoluto. Esto resultaba chocante para el jugador, ya que no es lo esperando. El jugador esperaba una respuesta por parte del NPC ante su acción.

Es por ello, que se quiso realizar este proyecto, que se centra en facilitar a los desarrolladores la creación de un NPC dotado de interactividad, presencia social y realismo.

## 4.3. Herramientas utilizadas

Para poder implementar la metodología explicada anteriormente, se ha hecho uso de diferentes aplicaciones y herramientas que nos permiten cumplir distintos aspectos como por ejemplo registrar las tareas y ponderarlas, subir las actualizaciones del proyecto o mantener la comunicación con el tutor para establecer las reuniones quincenales.

### Github

GitHub (Microsoft, 2008). Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. En ella tenemos subido el proyecto de Unity y ahí se actualizan todos los cambios que vamos haciendo a lo largo del desarrollo. El proyecto puede encontrarse en el siguiente enlace: `https://github.com/SeppukuAK/TFGCastilloSerrano`.

Para no tener que usar la consola bash de git, hemos utilizado GitKraken (Axosoft, 2019) como interfaz gráfica, que además dispone de integración con GitHub. De esta manera, nos resulta más sencillo realizar las tareas más básicas como crear repositorios, conectarlos con github y subir o bajar archivos.

### Slack

Slack (Slack Technologies, 2013). Es una herramienta de comunicación en equipo. Dispone de salas de chat organizadas por temas. Utilizada para comunicarse con el tutor, para preguntar dudas y organizar las reuniones.

### Trello

Trello (Atlassian, 2011). Es un software de administración de proyectos con una interfaz web. Utiliza un sistema de tarjetas virtuales para organizar las tareas, además de un sistema de puntos para poder valorarlas. Un punto de una tarea equivale a una hora realizada por un integrante del equipo.

### Drive

Drive (Google, 2012). Se trata de un servicio de alojamiento de archivos compartido entre los integrantes del equipo. Lo hemos utilizado para almacenar documentos, modelos, música, etc.

Además, hemos hecho uso de Google Docs y Slides, herramientas integradas en Drive, que hemos utilizado para la creación y edición de documentos

como por ejemplo el GDD, y creación de presentaciones PowerPoint, como las presentaciones realizadas para las reuniones generales.

### Overleaf

Overleaf (WriteLaTeX Limited, 2012). Es un editor de LaTeX online. LaTeX es un sistema de composición de textos, orientado a la creación de documentos. Lo hemos utilizado para la construcción de la memoria del TFG.

## 4.4. Test A/B

Una vez se han realizado todas las especificaciones anteriores, realizaremos unos test A/B con usuarios reales. Este consistirá en la prueba de dos grupos diferentes de usuarios del escenario de demostración de la herramienta, en forma de minijuego, con variaciones en el comportamiento del NPC instructor:

- **Usuario del grupo A**. Jugará a la demo con un NPC genérico imitando el comportamiento común en videojuegos convencionales. No usará las tareas y comportamientos construidos en la herramienta.
- **Usuario del grupo B**. Jugará a la demo con un NPC interactivo construido con la herramienta desarrollada, con acciones y condiciones que mejoran la sensación de presencia del usuario.

Una vez concluida la prueba, los usuarios rellenarán un cuestionario. Este contendrá preguntas acerca de sus sensaciones de presencia dentro de la prueba, especialmente con respecto al NPC, así como preguntas acerca de la comprensión de lo que ocurre en el juego. Para aportar rigor a esta evaluación, utilizaremos alguno de los siguientes formularios estándar utilizados para medir la presencia: Slather-Usoh-Steed (SUS) y Temple Presence Inventory (TPI).

Los objetivos de estas pruebas son los de encontrar que la utilización de la herramienta permite efectivamente tener NPCs que proporcionan una mayor sensación de presencia social al jugador de videojuegos en RV.

Capítulo 5

# Análisis, diseño e implementación

A lo largo de este capítulo se abordará el análisis, diseño y la implementación de tanto la herramienta como de la demostración especificadas en el capítulo 3. La herramienta ha sido nombrada como ***Social Presence VR*** y el juego que sirve como demostración de la herramienta ***Escape Room SP (Social Presence)***.

En el apartado de análisis y diseño y en el de implementación se establecerá una división entre lo relacionado con la herramienta y lo relacionado con la demostración. Además, dentro de todas estas secciones, siguiendo el esquema establecido en la especificación, se cubrirán los siguientes apartados:

- **Acciones.**
- **Condiciones.**
- **Conjuntos de comportamientos.**

## 5.1. Análisis y diseño

A continuación se explicará cómo ha sido el proceso de análisis y diseño de los diferentes aspectos del trabajo realizado.

### 5.1.1. Herramienta

En esta sección, trataremos el análisis y diseño de la propia herramienta a partir de su especificación.

#### 5.1.1.1. Árboles de comportamiento

El funcionamiento general de estos árboles (véase Figura 5.1) ha sido explicado en el capítulo 2, por lo que, en este apartado nos centramos en explorar cómo la utilización de estos árboles permite construir nuestra herramienta.

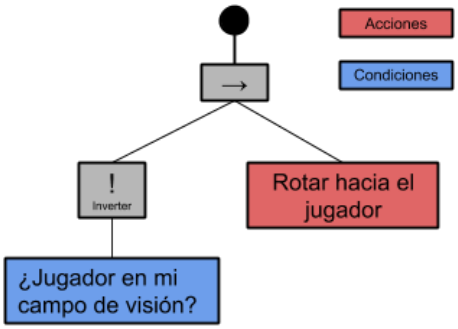



Figura 5.1: Ejemplo de un árbol de comportamiento sencillo: si el agente no ve al jugador, se rota

Tal y como se especificó anteriormente, nuestra herramienta funcionará como un **paquete de tareas y conjuntos de comportamientos** integrados en algún sistema de árboles de comportamiento.

Las **tareas** actuarán como **nodos** de árboles de comportamiento dentro de la herramienta y los **conjuntos de comportamiento** serán **árboles de comportamiento externos**.

Estos recursos de árboles de comportamiento, al mismo tiempo que proporcionan un gran conjunto de tareas, permiten a los desarrolladores crear las suyas propias. Esto es posible gracias a la utilización de una **API** (del inglés Application Programming Interface) con los métodos imprescindibles representados en la Figura 5.2.

```
// OnAwake is called once when the behavior tree is enabled. Think of it as a
constructor.
void OnAwake();

// OnStart is called immediately before execution. It is used to setup any
variables that need to be reset from the previous run.
void OnStart();

// OnUpdate runs the actual task.
TaskStatus OnUpdate();

// OnEnd is called after execution on a success or failure.
void OnEnd();
```

Figura 5.2: Métodos principales de la API de tareas de árboles de comportamiento

A la hora de añadir una nueva tarea, ya sea acción o condición, el usuario podrá encontrar todas las tareas desarrolladas en la herramienta dentro de un apartado que llamaremos *SocialPresenceVR*, respetando así la especificación del manejo de la interfaz establecido.

No sólo se ofrecen condiciones y acciones que se pueden añadir de forma independiente, sino que también crearemos un conjunto de comportamientos que están formados por condiciones y acciones que son dependientes entre sí. Estos conjuntos de comportamientos son árboles externos que se pueden ir añadiendo a cualquier parte del árbol general según las consideraciones de los desarrolladores.

#### 5.1.1.2. Acciones

A continuación, se analizan y diseñan las **acciones** especificadas:

##### Estar quieto

Devuelve *Running* continuamente hasta que sea interrumpido, generalmente por un *Conditional abort*. Tarea comúnmente conocida como ***idle***.

##### Esperar

Espera durante el tiempo establecido. Devuelve *Running* hasta que acabe la espera. Devolverá *Success* cuando haya transcurrido el tiempo. Parámetros:

- **Tiempo de espera**. Cantidad de tiempo en segundos a esperar.

##### Acciones corporales

Para dar expresividad al NPC, lo fundamental es que este sea capaz de **reproducir animaciones** (véase Figura 5.3), de hecho, la mayoría de las reacciones se van a representar de esta manera.

Nuestro diseño, tiene que garantizar la **simplicidad**, para que el usuario pueda crear multitud de animaciones sencillamente, únicamente utilizando los nodos ofrecidos y sin tener que utilizar *scripts* externos que el usuario tenga que modificar cada vez que quiera añadir una nueva animación.

Debido a que el usuario no tiene que preocuparse de la máquina de estados y modificar parámetros para transitar de un estado a otro, la herramienta se encargará de crear los estados y las transiciones. Por tanto, la forma del

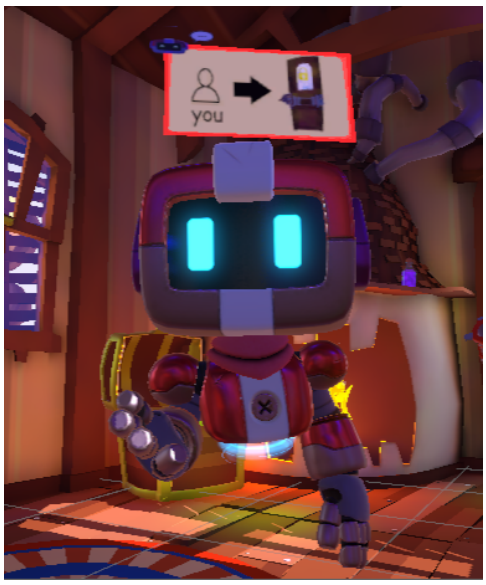



Figura 5.3: El NPC reproduciendo una animación de hablar mientras muestra un cartel

usuario de reproducir cualquier animación será mediante alguna de las dos siguientes acciones:

- **Reproducir animación**. Reproduce la animación una única vez. Retorna *Running* mientras se reproduce y devolverá *Success* cuando haya terminado.
- **Reproducir animación en bucle**. Reproduce la animación infinitamente hasta que sea interrumpida. Devuelve *Running* continuamente.

Parámetros comunes a ambas tareas:

- **Animación**. Animación a reproducir.
- **Duración de la transición**. Tiempo de transición entre la animación anterior y la actual.

#### Comunicación sonora

Para la comunicación, así como para la reacción sonora ante acciones del jugador, se plantean las dos siguientes tareas:

- **Reproducir sonido**. Empieza la reproducción de un sonido, devolviendo *Success* nada más comenzar su reproducción.

- **Reproducir sonido en bucle**. Reproduce un sonido indefinidamente, devolviendo *Running* hasta que sea interrumpido.

Parámetros de ambas tareas:

- **Fuente de audio**. Fuente de audio que tiene asociado el sonido a reproducir, así como los parámetros propios del sonido: volumen, prioridad, *pitch*, etc.

Devolverán *Failure* en caso de error por no tener asociado el parámetro.

#### Efectos especiales

Para la visualización de efectos especiales se plantea la tarea de **Reproducir partículas**. Empieza la reproducción de un sistema de partículas, devolviendo *Success* nada más comenzar su reproducción. Parámetros:

- **Sistema de partículas**. Sistema de partículas a reproducir.

Devolverán *Failure* en caso de error por no tener asociado el parámetro.

#### Movimiento

Primeramente, para tener la posibilidad de desplazar al NPC de un punto a otro inmediatamente se ha definido la siguiente acción:

- **Teletransportar un agente a un punto**. El agente se traslada al punto establecido. Retorna *Success* inmediatamente si no ha habido error, en cuyo caso devuelve *Failure*. Parámetros:
  - **Agente**. Referencia al NPC.
  - **Posición**. Posición a la que se teletransporta el NPC.

Existen muy variadas acciones de movimiento continuo. Las cuales compartirán los siguiente parámetros:

- **Velocidad**. Velocidad del agente.
- **Velocidad angular**. Velocidad angular del agente.
- **Distancia de llegada**. El agente ha completado la tarea cuando la distancia con el objetivo es menor o igual que el valor especificado.
- **Objetivo**. Objetivo sobre el que se realiza la acción.

Las acciones de movimiento consideradas más relevantes para el desarrollo de un NPC interactivo son las siguientes:

- **Desplazarse hacia un punto**. El NPC se desplaza hacia el punto establecido utilizando la malla de navegación. Retorna *Running* mientras se desplaza y *Success* cuando haya alcanzado el destino.
- **Seguir al objetivo**. El NPC sigue a un objetivo en movimiento utilizando la malla de navegación. Retorna *Running* continuamente, puesto que el agente tiene que continuar siguiendo al objetivo aunque haya alcanzado la distancia de llegada. Devolverá *Failure* en caso de error. Parámetros adicionales:
  - **Distancia de movimiento**. Empieza a seguir de nuevo al objetivo si la distancia es mayor que la establecida.
- **Evadir al objetivo**. El NPC evade al objetivo utilizando la malla de navegación. Retorna *Running* mientras se desplaza y *Success* cuando se haya alejado lo suficiente. Parámetros adicionales:
  - **Distancia de evasión**. El agente ha conseguido evadir al objetivo cuando la distancia es mayor que la establecida.

#### Rotación

Se han planteado dos acciones de rotación diferentes:

- **Rotación hacia un punto**. El NPC se rota hacia el punto establecido. Devuelve *Success* cuando haya acabado de rotarse y *Running* durante el proceso.
- **Dar la espalda al jugador**. El NPC se rota hasta dar la espalda al jugador. Devuelve *Success* cuando haya acabado de rotarse y *Running* durante el proceso.

Los parámetros son comunes a ambas acciones:

- **Objetivo**. Objetivo hacia el que se rota el NPC.
- ***Epsilon***. El NPC ha acabado de rotar cuando el ángulo de diferencia es menor que el establecido.
- **Velocidad angular**. Velocidad a la que se rota el NPC.

#### Interacción con objetos

El NPC es capaz de interactuar con los objetos de la escena para ayudar al jugador. Dispone de las siguientes acciones:

- **Recoger objeto**. El NPC coge un objeto interactuable con la mano. Devuelve *Success* si no ha habido error al tratar de coger el objeto, en cuyo caso devolverá *Failure*. Parámetros:
  - **Objeto interactuable**. Objeto interactuable a recoger.
- **Entregar objeto**. El NPC ofrece un objeto al jugador. Retornará *Running* mientras está ofreciéndolo y devuelve *Success* cuando el jugador agarre el objeto. Retornará *Failure* en caso de error. Parámetros:
  - **Objeto interactuable**. Objeto interactuable a recoger.
  - **Jugador**. Referencia al jugador.

#### Enseñar cartel

El NPC enseña un cartel al jugador durante el tiempo establecido. Utilizado para dar información al jugador mediante lenguaje no verbal. Retorna *Success* inmediatamente después de crear el cartel, siempre que no haya habido error, en cuyo caso devolverá *Failure*. Parámetros:

- **Cartel**. Referencia al cartel que se desea explicar.
- **Posición**. Posición en la que aparece.
- **Tiempo**. Tiempo en el que permanece activo el cartel.

#### Inicialización y referencias

Las tareas de esta categoría no han sido mencionadas en la especificación por tratarse de nodos auxiliares que solo se entienden en el concepto de la programación. Para el correcto funcionamiento de todo el diseño de la herramienta, son necesarias estas acciones auxiliares de inicialización y adquisición de referencias:

- **Inicializar Social Presence NPC**. Parámetros:
  - **Mano del NPC**. Referencia a la mano del NPC con la que coge objetos.
  - **Altura por defecto del jugador**. Altura que se tendrá en cuenta para considerar si el jugador está agachado o no.

    - **Resetear animaciones**. *Booleana* que indica si en esta ejecución se crean todas las transiciones entre estados de animación o se utiliza la creada anteriormente.

- **Obtener cabeza del jugador**. Obtiene la referencia a la cabeza del jugador. Parámetros:
    - **Cabeza del jugador**.
- **Obtener manos del jugador**. Obtiene la referencia a las manos del jugador. Parámetros:
    - **Mano izquierda del jugador**.
    - **Mano derecha del jugador**.

Todas estas tareas retornan *Success* inmediatamente si no ha habido error, en cuyo caso devolverán *Failure*.

#### 5.1.1.3. Condiciones

En cuanto a las tareas de **condiciones**, siguiendo la estructura ya establecida en la especificación, podemos clasificarlas en los siguientes categorías:

##### Condiciones corporales

Tal y como se especificó anteriormente, este tipo de condiciones pertenecen al campo de la **Kinesia y Proxémica**, exceptuando los gestos que se hacen con las manos:

- **Ver al jugador**. Devuelve *Success* si se encuentra a la vista el jugador con la distancia y ángulo establecidos (véase Figura 5.4), en caso contrario devolverá *Failure*. Esta condición es relevante para mantener un comportamiento respetuoso con las distancias proxémicas. Parámetros:
    - **Jugador**. Referencia a la cabeza del jugador.
    - **Distancia de vista**. Distancia a la que el NPC es capaz de ver al jugador.
    - **Ángulo de vista**. Ángulo del campo de visión del NPC en el que puede ver al jugador.
- **Jugador mira al NPC**. Devuelve *Success* si el jugador tiene en su campo de visión al NPC, en caso contrario devolverá *Failure*. Esta condición es importante para ser capaz de distinguir si el jugador está tratando de interactuar con el NPC. Parámetros:

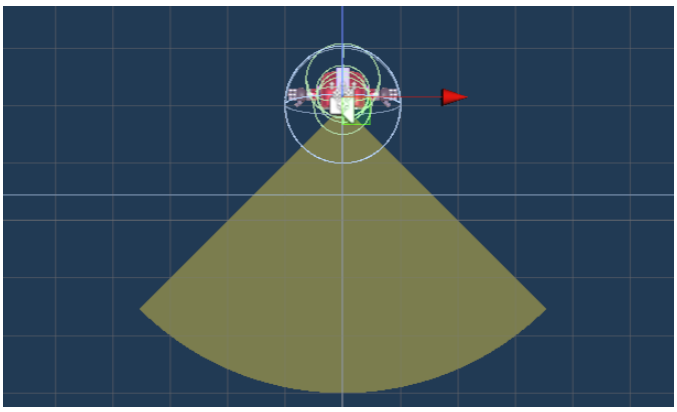


Figura 5.4: Representación del campo de visión en la condición 'Ver al jugador'

  - **Jugador**. Referencia a la cabeza del jugador.
  - **Ángulo de vista**. Ángulo del campo de visión del jugador en el que puede ver al NPC.

- **Jugador mira fijamente a un objeto**. Devuelve *Success* si el jugador mira fijamente al objeto durante el tiempo establecido, en caso contrario, devolverá *Failure*. Esta condición cumplida durante mucho tiempo, puede tener la reacción de hacer sentir incomodo al NPC. Parámetros:
  - **Jugador**. Referencia a la cabeza del jugador.
  - **Objeto**. Referencia al objeto que se desea comprobar si está mirándolo.
  - **Distancia de vista**. Distancia a la que el jugador es capaz de ver al objeto.
  - **Duración**. Tiempo que tiene que estar mirando al objeto para que devuelva *Success*.
- **Jugador da la espalda al NPC**. Devuelve *Success* si el jugador está dando la espalda al NPC, dando a entender que el jugador no está prestando atención al mismo. En caso contrario, devolverá *Failure*. Parámetros:
  - **Jugador**. Referencia a la cabeza del jugador.
- **Jugador quieto**. Devuelve *Success* si el jugador se mantiene quieto durante mucho tiempo, indicador de que el jugador puede encontrarse un poco perdido. En caso contrario, devolverá *Failure*.

- **Jugador está agachado**. Devuelve *Success* si el jugador se encuentra agachado. En caso contrario, devolverá *Failure*. Que una persona esté agachada implica que algo se le ha caído o algo ha perdido, momento relevante a detectar para que el NPC le pueda ayudar. Parámetros:
  - **Porcentaje de agachado**. Porcentaje de altura actual con respecto a la del jugador levantado en la que se considera que el jugador está agachado.

#### Interacción del jugador con el entorno

Condiciones que detectan cuando el jugador interactúa con los diferentes objetos que hay en la escena. Los más relevantes, ya especificados anteriormente, son los siguientes:

- **Jugador coge un objeto**. Devuelve *Success* cuando el jugador está agarrando un objeto con alguna de las manos. En caso contrario retornará *Failure*. Parámetros:
  - **Objeto interactuable**. Objeto especificado con el que tiene que interactuar para que se cumpla la condición.
- **Jugador usa un objeto**. Devuelve *Success* cuando el jugador está usando un objeto con alguna de las manos. En caso contrario retornará *Failure*
  - **Objeto interactuable**. Objeto especificado con el que tiene que interactuar para que se cumpla la condición.
- **Jugador señala a un objeto**. Devuelve *Success* cuando el jugador está señalando un objeto con alguna de las manos. En caso contrario retornará *Failure*.
  - **Objeto interactuable**. Objeto especificado con el que tiene que interactuar para que se cumpla la condición.
- **Jugador se teletransporta**. Devuelve *Success* cuando el jugador se teletransporta a cualquier de los nodos en la escena. En caso contrario retornará *Failure*.

#### Interacción física del jugador con el NPC

Conjunto de condiciones que se basan en la colisión de algo con el NPC, siendo las más relevantes:

- **Ser tocado por el jugador**. Retorna *Success* cuando el jugador le toca con la parte del cuerpo especificada. En caso contrario, devolverá *Failure*. Parámetros:
  - **Parte del cuerpo**. Referencia a la mano o cabeza del jugador.
- **Ser golpeado por un objeto**. Devuelve *Success* cuando detecta que ha sido golpeado por algún objeto lanzado por el jugador.
- **Ser golpeado por el jugador**. Retorna *Success* cuando el jugador le golpea con la mano especificada. En caso contrario, devolverá *Failure*. Parámetros:
  - **Mano del jugador**. Referencia a la mano del jugador
  - **Velocidad**. Velocidad de la mano al colisionar con el NPC para que sea considerado como golpe.

#### Gesticulación del jugador

En este apartado se contemplan las condiciones, ya especificadas anteriormente, de gestos que pertenecen a la **Kinesia** que requieren de las manos o la cabeza para interactuar y no requieren de contacto físico:

- **Gestos con la cabeza**. Condiciones utilizadas para detectar si el jugador entiende o está de acuerdo con el NPC:
  - **Asentir**. Retorna *Success* cuando el jugador asiente con la cabeza, *Failure* en caso contrario.
  - **Negar**. Retorna *Success* cuando el jugador niega con la cabeza, *Failure* en caso contrario.
- **Gestos con las manos**. Diversas condiciones que requieren de gesticulación con las manos con diferentes significados:
  - **Saludar**. Retorna *Success* cuando se ha detectado un gesto de saludar con la mano, *Failure* en caso contrario.
  - **Señalar**. Retorna *Success* cuando el jugador señala con el dedo índice al objeto especificado, *Failure* en caso contrario. Parámetros:
    - **Objeto**. Objeto a detectar si está siendo señalado por el jugador.
    - **Distancia**. Máxima distancia a la que se puede encontrar el objeto para considerar que el jugador lo está señalando.

- **Levantar la mano**. Devuelve *Success* cuando el jugador levanta una mano, como un gesto de pedir ayuda, y *Failure* en caso contrario. Parámetros:
  - **Diferencia**. Diferencia de distancia entre la mano y la cabeza, para que se considere que la mano está siendo levantada como un gesto de pedir ayuda.

#### 5.1.1.4. Conjuntos de comportamientos

A continuación se expone el diseño de los conjuntos de comportamientos, basándose en la especificación.

- ***Idle***. Comportamiento que realiza el NPC cuando no se cumple ninguna condición a realizar. Mantiene una animación en bucle mientras mira al jugador constantemente.
- **Rotar hasta encontrar al jugador**. Mientras no se encuentre el jugador a la vista, se rota hasta encontrarle. Este comportamiento es utilizado en diferentes conjuntos de comportamientos.
- **Buscar objeto y recogerlo**. Si el NPC tiene que recoger un objeto, se desplaza hacia él y lo recoge.
- **Buscar jugador y entregar objeto**. Si el NPC tiene que entregar un objeto al jugador, le sigue y se lo ofrece hasta que lo recoja.
- **Recoger y entregar un objeto al jugador**. Conjunción de los conjuntos de comportamiento de recoger y entregar el objeto establecido.
- **Hablar**. El NPC tiene su propio idioma y lo usará mientras da algún tipo de explicación sobre alguna tarea, o como reacción a alguna acción causada por el jugador. Para comunicarse reproducirá sonidos y onomatopeyas.
- **Explicar cartel**. Mostrar el cartel establecido mientras reproduce una animación y habla.
- **Acercarse al jugador cuando se ha teletransportado**. Cuando el jugador usa el teletransporte puede alejarse demasiado y con este comportamiento el NPC se acercaría hacia él. Reproduce una animación con un movimiento más veloz que el de andar.
- **Enfadarse al ser golpeado por el jugador**. Cuando el NPC recibe alguna colisión por un objeto lanzado por el jugador, reproduce una animación de enfado.

- **Saludar cuando el jugador saluda**. Cuando jugador saluda con la mano al NPC, este le responde reproduciendo una animación de saludo.
- **Comportamiento proxémico**. Este comportamiento trata de mantener siempre una distancia social con el jugador y reacciona cuando se acerca o aleja del limite. Este a su vez, está formado por conjuntos de comportamientos:
  - **Colisión física con el jugador**. Detecta que el jugador se encuentra colisionando y se aleja de él reproduciendo alguna animación de susto mientras le mira.
  - **Distancia física con las manos del jugador**. Detecta que las manos del jugador se encuentran colisionando y se aleja de él reproduciendo alguna animación de enfado mientras le mira.
  - **Distancia mínima con el jugador**. Detecta que el cuerpo del jugador se encuentra demasiado cerca y lo evade reproduciendo alguna animación preocupación.
  - **Distancia mínima con las manos del jugador**. Detecta que las manos del jugador se encuentran demasiado cerca y reproduce una animación de preocupación.
  - **Distancia media con el jugador**. Detecta que el cuerpo del jugador se encuentra demasiado cerca y reproduce una animación de guardia.
  - **Distancia social con el jugador**. Detecta que el jugador se encuentra a una distancia mayor de la distancia social y se acerca hacia él reproduciendo una animación de moverse.

#### Ejemplos visuales de conjuntos de comportamientos

A continuación se van a mostrar el diseño de algunos de los conjuntos de comportamiento representados mediante árboles de comportamiento que nos han resultado los más interesantes de enseñar:

- **Buscar jugador y entregar objeto.**

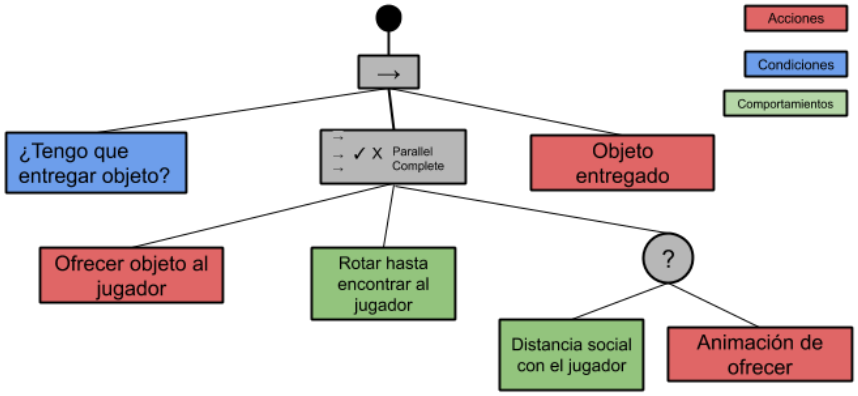



Figura 5.5: Árbol de comportamiento de buscar jugador y entregar objeto

- **Explicar cartel**.

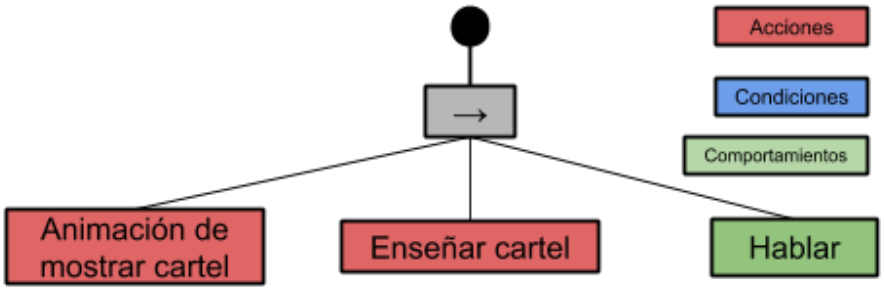



Figura 5.6: Árbol de comportamiento de explicar cartel

- **Comportamiento proxémico.**

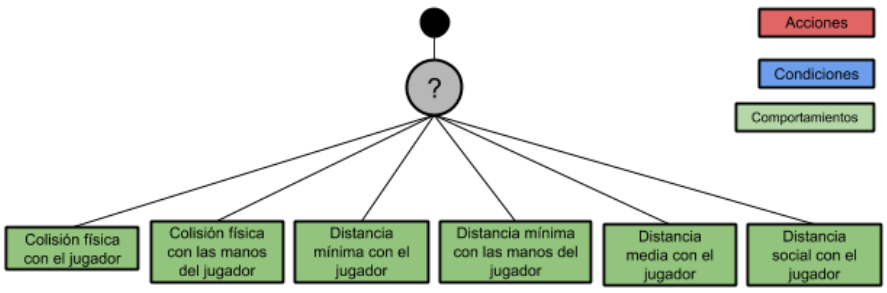



Figura 5.7: Árbol de comportamiento de comportamiento proxémico

- **Distancia social con el jugador.**

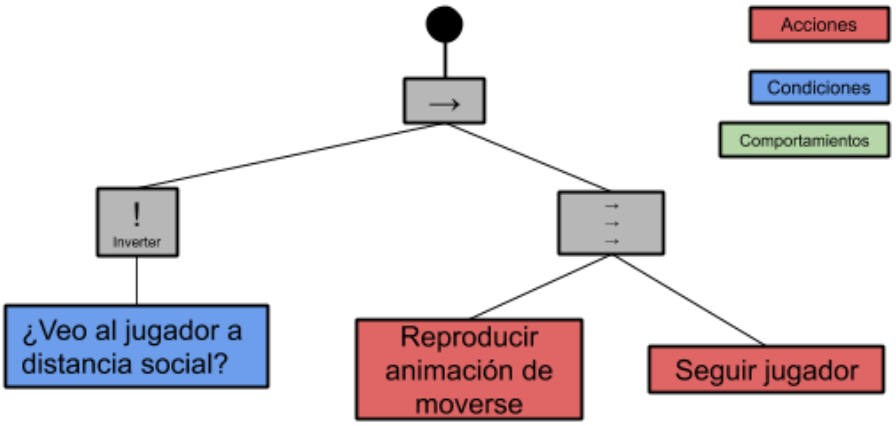



Figura 5.8: Árbol de comportamiento de distancia social con el jugador

### 5.1.2. Demostración

A continuación, se tratará el análisis y diseño de la escena de demostración creada para implementar la herramienta.

#### 5.1.2.1. Descripción de la escena de ejemplo

Se trata de una escena en la cual el jugador se encuentra en una casa y su objetivo es lograr salir de ella. Para ello, cuenta con la ayuda de ***Jammo***, un robot que hará de guía y te irá ayudando. Durante su estancia allí, las acciones del jugador repercutirán en el comportamiento del robot.

Se trata de una pequeño juego experimental, para estudiar si un NPC dotado de presencia social tiene un impacto notable en el jugador, frente a un NPC más pasivo que no interactúa con él y que apenas reacciona a sus acciones.

El videojuego dispone de dos variantes:

- **Variante A**. El robot con el que te encuentras en la casa es un NPC más pasivo, que se limita a darte la primera explicación y a partir de ahí lo único que hará será seguirte, imitando el comportamiento de la mayoría de los videojuegos.
- **Variante B**. El robot con el que te encuentras en la casa es un NPC más activo, diseñado para suscitar la sensación de presencia social, que estará pendiente del jugador, reaccionará a todas sus acciones y

le irá ayudando en las distintas etapas que hay hasta poder salir de la casa. Además, si el jugador se pierde en algún momento durante la experiencia, siempre puede levantar la mano para que el robot le vuelva a indicar qué es lo que debe hacer a continuación.

#### 5.1.2.2. Aspectos generales

En esta escena de ejemplo el objetivo es tener un **escenario** (véase Figura 5.9) lo suficientemente **inmersivo** como para que el usuario reaccione de la misma manera que actuaría en la vida real. A pesar no haber comunicación oral entre el jugador y el robot, se ha creado un NPC que actúa de forma creíble frente a las posibles acciones del jugador.

Figura 5.9: Imagen del escenario de la escena de ejemplo

Para buscar una experiencia inmersiva, los movimientos del jugador se ven reflejados en el avatar. Esto condiciona a que el espacio de movimiento sea un área reducida.

A pesar de esto, el jugador podrá visitar las zonas más relevantes de la casa (que es bastante más grande que el espacio de movimiento del jugador) gracias al teletransporte. El control de los mandos se basa en acciones básicas humanas (coger, lanzar e interactuar con objetos) buscando ser intuitivos. Evitamos el uso de botones y joysticks, salvo para acciones concretas como atraer un objeto lejano o teletransportarse, que son acciones especiales que sí necesitan de botones/joystick. El juego trata de conseguir la coherencia en la interacción del jugador con los diferentes objetos y posibles acciones que puede realizar.

Se ha tenido especial cuidado en hacer saber al jugador cuándo ha hecho algo bien y cuándo ha hecho algo mal, y esto se puede ver gracias a las

animaciones y a la expresión en la cara (los ojos) del robot, que hará una animación triste si el jugador falla en alguna tarea, y se pondrá contento o aplaudirá si el jugador ha hecho la tarea correctamente.

Los signos (véase Figura 5.10) en este juego van a ser muy importantes para mantener la comunicación entre el jugador y el robot. Serán lo suficientemente claros y básicos como para que cualquier persona de cualquier edad pueda interpretarlos de manera correcta.

Figura 5.10: Imagen de un cartel indicando que se puede levantar la mano

Se han establecido comportamientos en el robot para detectar acciones que son estándar en las relaciones interpersonales como por ejemplo la distancia proxémica. Por ejemplo, el robot se sentirá incómodo cuando el jugador se acerca demasiado hacia él o se si le tocas, ya que la relación entre ambos es una relación entre dos personas desconocidas, por lo que no hay confianza como tal.

#### 5.1.2.3. Estructuración de la escena de ejemplo

Al tratarse de un juego experimental para comprobar la utilidad de la herramienta que hemos realizado para dotar al robot de presencia social, no existe un trasfondo narrativo muy desarrollado, se trata simplemente de un minijuego en el que hay una serie de puzles a completar y una vez abierta la puerta de la casa, al salir de ella termina todo.

La escena de ejemplo no tiene distintos niveles o escenarios, pero se ha querido dividir en fases (véase Figura 5.11), útiles también a la hora de diseñar el comportamiento del robot.

Existen cuatro fases en juego, que son las siguientes:

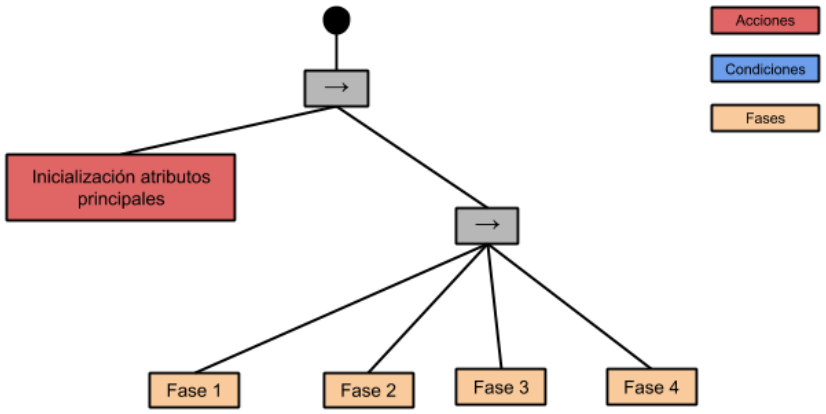



Figura 5.11: Diagrama del árbol principal del juego

**Fase 1**

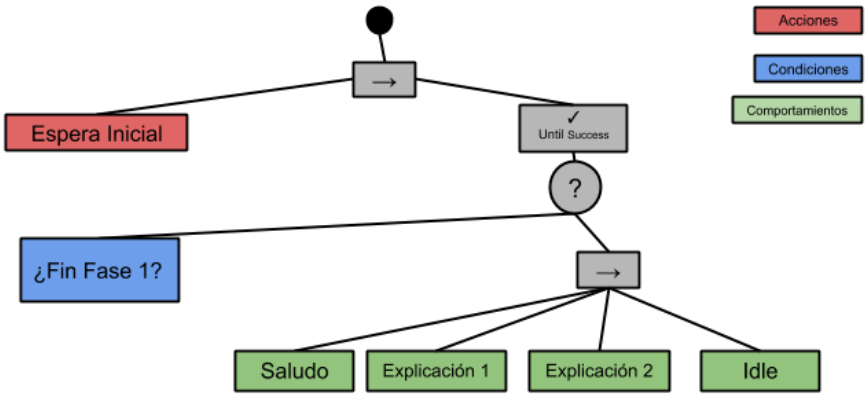



Figura 5.12: Diagrama del árbol de Fase 1

Primera fase del juego en la cual el robot espera unos segundos antes de comenzar. Una vez acabada la espera, el robot saluda al jugador y se dispone explicarle cuál es su objetivo principal.

Esperará hasta que el jugador asienta con la cabeza para dar a entender que ha comprendido lo que dice o negar con la cabeza para dar a entender que no ha comprendido.

Si el jugador no ha comprendido, el robot se dirigirá hacia la puerta y la señalará, dando a entender al jugador que tiene que salir por esa puerta.

Si ha comprendido, el robot le explicará cual es la primera tarea a hacer, que es poner el libro en el atril y también esperará respuesta por parte del jugador. Si el jugador no ha entendido esta tarea, el robot señalará a la llave, para indicar al jugador que debe usarla para encontrar el libro del atril.

Para poder realizar la tarea, el jugador tiene que coger la llave que está en la mesa y ponerla sobre el cajón con cerradura, abriendo así el cajón. Dentro del mismo estará el libro, que el jugador deberá coger y colocarlo sobre el atril. Una vez colocado, se abrirá, se activarán los teletransportes y el libro mostrará los controles de cómo teletransportarse. El robot se alegrará de que el jugador haya completado esta tarea y se dará paso a la siguiente fase.

**Fase 2**

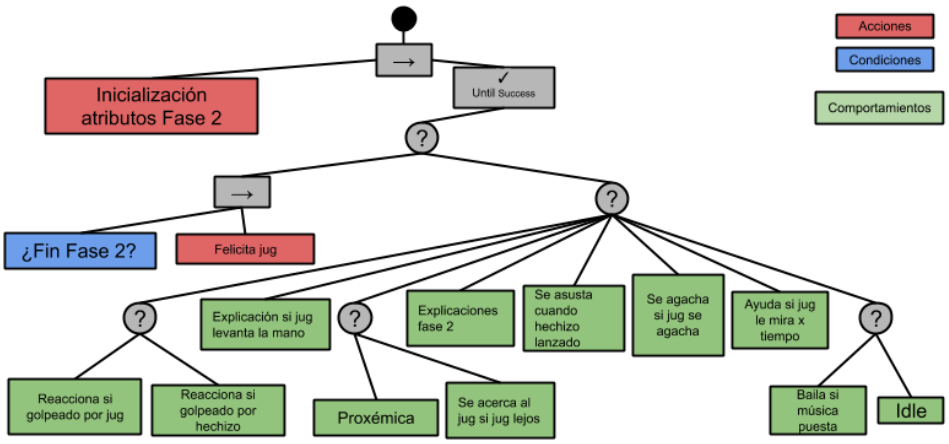



Figura 5.13: Diagrama del árbol de Fase 2

Una vez colocado el libro en el atril, se dará paso a la segunda fase. En esta fase, el robot explicará al jugador que necesita abrir el caldero y para ello necesita la varita.

A partir de esta fase, el jugador podrá levantar la mano en caso de no saber lo que tiene que hacer en ese momento. Si el jugador se queda mirando durante un tiempo al robot, este le mostrará un cartel indicando que puede levantar la mano en caso de tener dudas.

Si el jugador levanta la mano en este momento, el robot o bien señalará a la varita, recordando al jugador que debe usarla, o bien volverá a mostrar el cartel del caldero y la varita.

Cuando el jugador coja la varita, el robot le mostrará un cartel con información sobre el uso de la varita.

Si en este momento el jugador levanta la mano, el robot o bien se dirigirá al caldero para recordar al jugador que tiene que abrir el caldero, o bien mostrará de forma aleatoria cualquiera de los siguientes carteles: el de uso de la varita o el de la varita y el caldero.

Una vez el jugador dispare con la varita hacia la diana que hay en la tapa del caldero, se dará paso a la siguiente fase.

### Fase 3

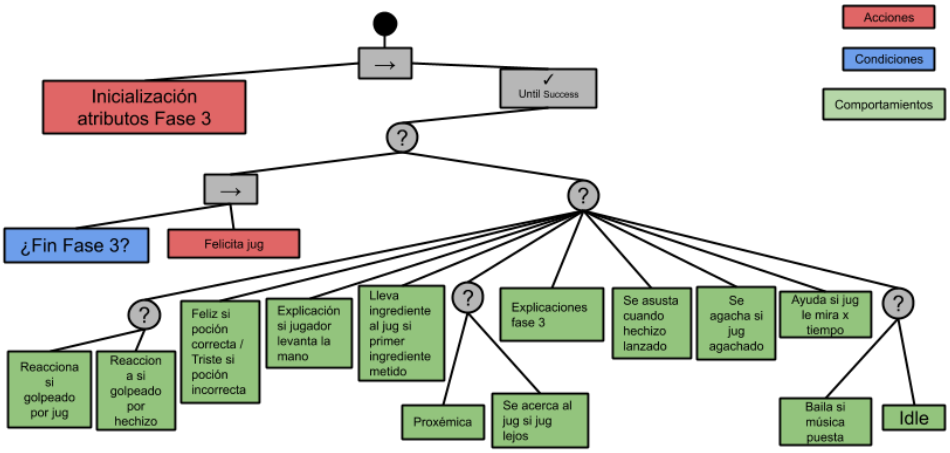



Figura 5.14: Diagrama del árbol de Fase 3

Nada más comenzar esta fase, el robot se pondrá contento porque el jugador ha abierto la tapa del caldero. A continuación, procederá a explicar la siguiente tarea a realizar, que es colocar los ingredientes que marca la receta en el caldero y establecer los parámetros correctos para poder crear la poción.

Si en este momento el jugador levanta la mano, el robot o bien señalará al caldero recordando así al jugador que tiene introducir los ingredientes en el caldero, o bien volverá a mostrar el cartel con la receta.

Una vez que el jugador meta un ingrediente en el caldero, ya sea correcto o no, el robot irá a buscar uno de los ingredientes correctos necesarios y se lo ofrecerá al jugador para ayudarle y agilizar el trabajo.

Si el jugador ha metido algún ingrediente incorrecto, o algunos de los parámetros para hacer la poción es incorrecto, se creará una poción con una imagen de una X mostrando que la poción fabricada es incorrecta, y el robot se pondrá triste, dando a entender que no lo ha hecho bien. El jugador deberá volver a repetir todo el proceso de meter los ingredientes en el caldero y establecer los parámetros.

Si el jugador ha seguido los pasos de la receta, se creará una poción que se mostrará como correcta. El robot aplaudirá al jugador y procederá a mostrar la siguiente tarea a realizar que es verter la poción sobre el colador de la puerta.

Si en este momento el jugador levanta la mano, el robot o bien volverá a mostrarle el mismo cartel, o se dirigirá a la puerta y la señalará, dando a entender que tiene que ir a la puerta para verte la poción.

Si el jugador vierte la poción sobre el colador de la puerta, esta se abrirá y se pasará a la última fase.

**Fase 4**

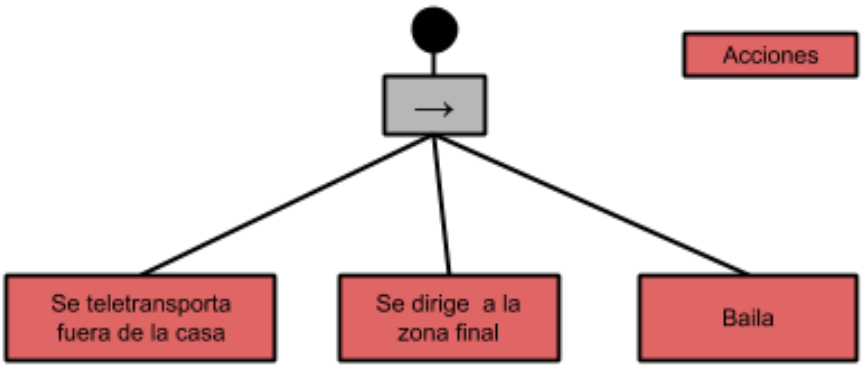



Figura 5.15: Diagrama del árbol de Fase 4

En esta última fase el robot se teletransportará fuera de la casa y se dirigirá al punto de teletransporte que hay fuera de la casa. Una vez llegue a dicho punto, una música empezará a sonar y el robot comenzará a bailar, celebrando que el jugador ha completado la prueba.

Una vez que el jugador se teletransporte fuera de la casa, el juego finalizará.

#### 5.1.2.4. Descripción del NPC

El único personaje aparte del jugador que se encuentra en la escena de ejemplo es Jammo (véase Figura 5.16), el robot que actuará como guía y ayudará al jugador a completar las diferentes tareas hasta salir de la casa, que es el objetivo final.

Se ha tratado de dotar de personalidad a Jammo, haciendo que muestre **diferentes emociones** como por ejemplo **alegría, asco, miedo o tristeza**,

emociones que se verán reflejadas a través de la actitud del robot y de sus ojos.

Jammo es un robot que estará pendiente del jugador en todo momento. Reaccionará a cada una de las acciones de este, le afecten o no a él, y ya sean positivas o negativas.

Figura 5.16: Imagen de Jammo saludando al jugador

A continuación, se procederá a indicar qué acciones, condiciones y conjuntos de comportamientos puede realizar Jammo a lo largo de la escena de ejemplo.

### Acciones

En este apartado se van a explicar las acciones que son específicas de la demostración.

- **Cambiar el color y forma de ojos del robot**. El robot dependiendo de la emoción que esté sintiendo en ese momento, tendrá unos ojos u otro. Cada emoción tiene un color y forma diferente. Devolverá *Success* una vez modificados los ojos. Parámetros:
  - **Estado de los ojos**. Estado de los ojos al que se quiere cambiar.
- **Cambiar la textura del robot**. El robot puede cambiar de textura y tener colores diferentes. Devolverá *Success* una vez modificada la textura. Parámetros:
  - **Lista de colores**. Secuencia de los colores en los que va a ir cambiando el robot cada vez que se active esta acción.

- **Encontrar ingrediente necesario**. El robot puede encontrar un ingrediente que forma parte de la receta dentro de un área establecida. Devuelve *Sucess* si existe un ingrediente dentro de dicha área y *Failure* en caso contrario. Parámetros:
  - **Distancia**. Distancia a la que se quiere comprobar si hay un ingrediente.
  - **Objeto interactuable a coger**. Ingrediente encontrado por el robot.

### Condiciones

Las condiciones de la escena de ejemplo se clasifican en los siguientes apartados:

- **Comprobar si el jugador ha abierto el libro**. Esta comprobación da pie al inicio de la Fase 2. Devuelve *Success* si el libro está abierto y *Failure* en caso contrario.
- **Comprobar si el jugador ha abierto el caldero**. Esta comprobación da pie al inicio de la Fase 3. Devuelve *Success* si el caldero está abierto y *Failure* en caso contrario.
- **Comprobar si el jugador ha abierto la puerta**. Esta comprobación da pie al inicio de la Fase 4. Devuelve *Success* si la puerta está abierta y *Failure* en caso contrario.
- **Comprobar si el caldero ya tiene el primer ingrediente en su interior**. Esto da a entender al robot que el jugador ha entendido que tiene que meter un ingrediente en el caldero. Devuelve *Success* si hay algún ingrediente dentro de la lista de ingredientes del caldero y *Failure* en caso contrario.
- **Comprobar si la radio está encendida** Se comprueba si el jugador ha encendido la radio y está sonando música en un volumen determinado. Devuelve *Success* si se ha encendido la música y se encuentra a un volumen igual o mayor al determinado, y *Failure* en caso contrario. Parámetros:
  - **Volumen**. Volumen en el cual se considera que la música está lo suficientemente alto como para reaccionar a ella.
- **Comprobar si el jugador ha creado la poción, ya sea de forma correcta o incorrecta**. El NPC reacciona ante el hecho de que el jugador haya logrado crear la poción tras haber seguido los pasos de la

receta. Devuelve *Success* si la poción ha sido creada y *Failure* en caso contrario. Parámetros:

- **Poción correcta**. Booleana para indicar si la poción creada es correcta o no.

### Conjuntos de comportamientos

- **Saludo inicial**. El NPC se acerca al jugador, se gira hacia él y reproducirá una animación de saludo.
- **Explicaciones iniciales**. Al inicio de la prueba, el robot explicará al jugador el objetivo final y la primera tarea que tiene que realizar el jugador. Entre cada explicación, el robot esperará una respuesta por parte del jugador. Tendrá que asentir o negar con la cabeza para demostrar si ha entendido o no lo que se le ha explicado.
- **Explicaciones posteriores**. A diferencia de las explicaciones iniciales, en este tipo de explicaciones el robot se limitará a dar la explicación pertinente, no esperará a que el jugador asienta o niegue con la cabeza, ya que al ser necesarias varias explicaciones a lo largo de la prueba, puede cortar mucho el ritmo y dinamismo de la experiencia.
- **Explicación en caso de que el jugador levante la mano**. Si el jugador se siente perdido y necesita que el robot le recuerde qué tiene que hacer en ese momento, puede levantar la mano para pedir ayuda. Una vez levantada, el robot volverá a explicar la tarea actual de nuevo o bien señalará el objeto relevante y necesario para realizar dicha tarea.
- **El robot se agacha si el jugador está agachado**. Si el jugador se agacha es porque ha perdido algo, por lo que el robot reproducirá la animación de agacharse también como dando a entender que está ahí para ayudarle.
- **El robot ayudará al jugador si le mira durante un tiempo**. Si el jugador se queda mirando al robot durante un rato es porque necesita algo de él o que puede tener alguna duda, por lo que en este caso si le mira, el robot le explicará que puede levantar la mano.
- **El robot baila si hay música puesta**. Si el jugador enciende la radio, el robot reproducirá una animación de mover la cabeza al ritmo de la música mientras la radio siga encendida.
- **El robot se molesta si es golpeado por el jugador**. Si un objeto colisiona con el jugador, el robot reproducirá una animación de molestia, y si la mano del jugador jugador colisiona con el robot, este huirá hacia atrás y reproducirá la animación de enfado mayor.

- **El robot cambia de color si es golpeado por un hechizo de la varita**. Si el jugador dispara con la varita al robot, este cambiará de textura y reproducirá una animación de sorpresa.
- **Aplaudir al jugador si ha creado la poción correcta**. El robot reproducirá la animación de aplaudir si el jugador ha conseguido seguir correctamente los pasos de la receta.
- **Mostrarse triste si el jugador ha creado la poción incorrecta**. Si el jugador falla a la hora de crear la poción, el robot reproducirá una animación de tristeza.
- **Mostrar la receta al jugador cada vez que se encuentra en la zona del caldero**. Para poder ayudar al jugador a que tenga presentes los ingredientes y los parámetros del caldero a establecer que tiene la receta y evitar que el jugador tenga que estar levantando la mano constantemente, ya que son cosas concretas, cada vez que se encuentre en la zona del caldero, el robot le mostrará el cartel de la receta de forma constante.
- **El robot ayuda al jugador después de colocar el primer ingrediente en el caldero**. Para agilizar la tarea al jugador, el robot se dirigirá a un ingrediente de los necesarios para la receta, reproducirá la animación de cogerlo, volverá adonde se encuentra el jugador y reproducirá la animación de entregar al ingrediente, esperando a que el jugador lo coja.

#### Ejemplos visuales de conjuntos de comportamientos

A continuación se van a mostrar tres ejemplos de conjuntos de comportamiento representados mediante árboles de comportamiento que nos han resultado los más interesantes de enseñar.

- **El robot ayuda al jugador después de colocar el primer ingrediente en el caldero**.

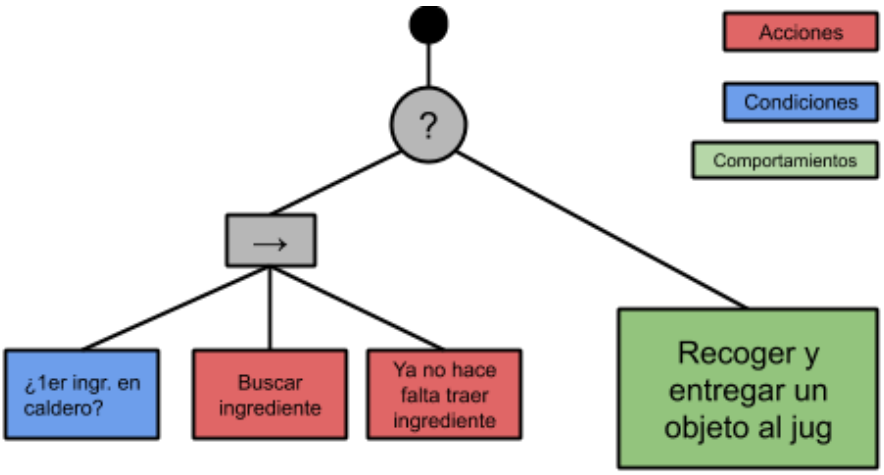



Figura 5.17: Árbol de comportamiento de ayuda al jugador tras colocar el primer ingrediente en el caldero

- **Explicación en caso de que el jugador levante la mano**.

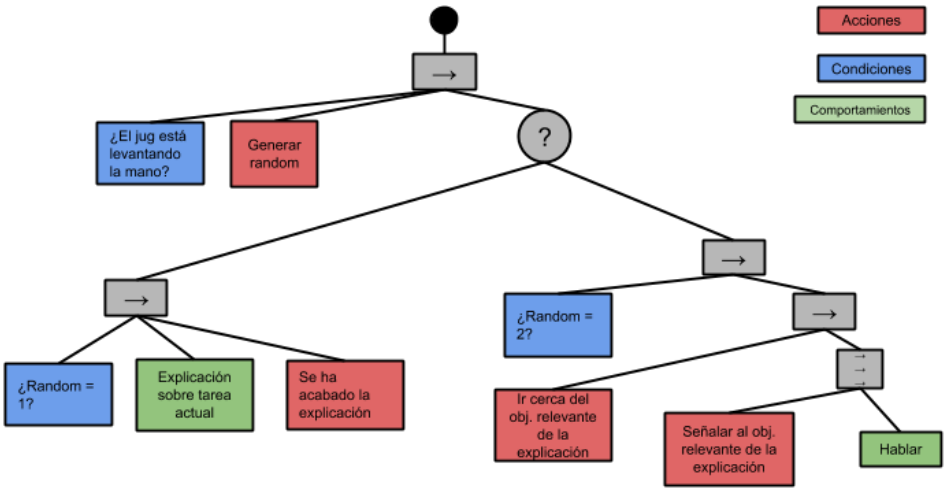



Figura 5.18: Árbol de comportamiento de explicación al levantar la mano

- **Explicaciones iniciales**.

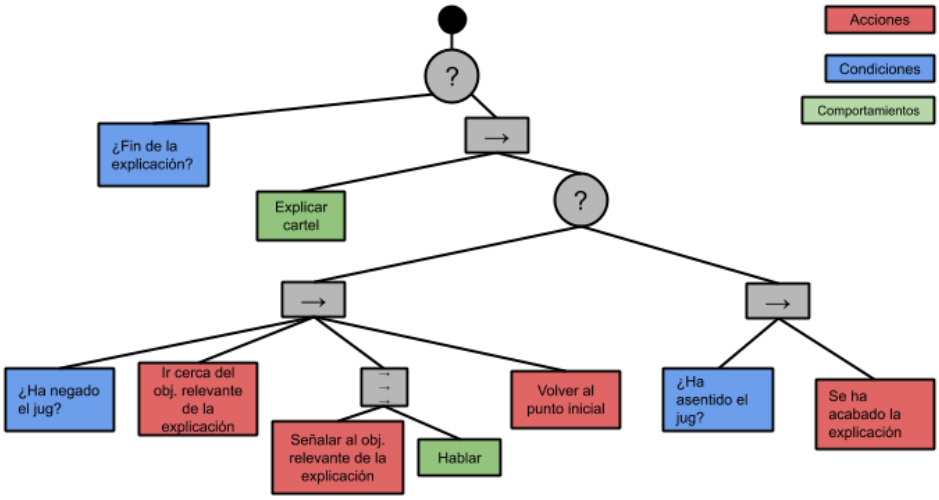



Figura 5.19: Árbol de comportamiento de explicación inicial

#### 5.1.2.5. Configuración

Todos los cambios en la configuración ya sea de las gafas o del sonido se realizan desde el propio menú que tienen integrado los dispositivos de RV, pues cada uno tiene su propio menú de configuración.

#### 5.1.2.6. Interfaz y control

En la escena original en la que nos hemos basado para el juego, existía un menú dentro de un reloj que posee el jugador en la muñeca, que mostraba la opción de resetear el juego. Como no funcionaba de manera correcta, se decidió eliminarlo.

Los controles tanto de mando (véase Figura 5.20) como gestos que se pueden realizar en el juego vienen explicados mediante los carteles que va mostrando el NPC a lo largo del juego y mediante unos pocos carteles que se encuentra in situ.

Los controles referentes al mando son los siguientes:

- **HandTrigger**. Agarrar un objeto
- **IndexTrigger**. Usar un objeto
- **Axis2D.Thumbstick**. Tiene tres posibles detecciones dependiendo de la acción que se realice sobre este:

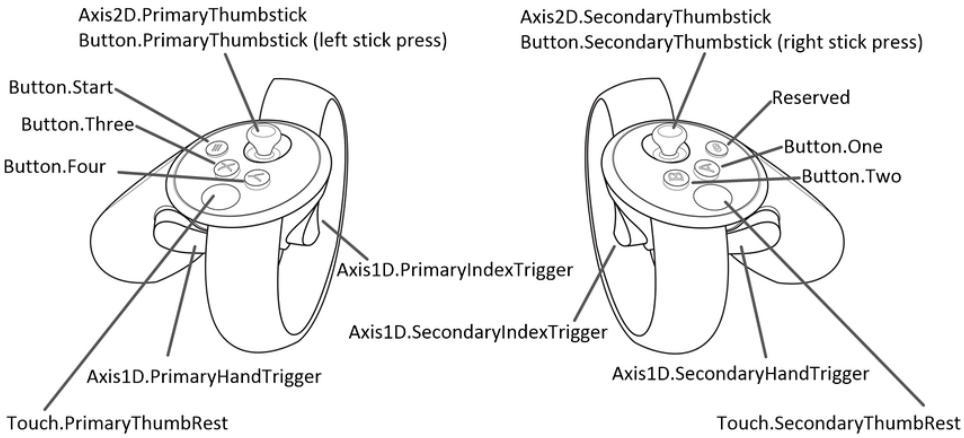



Figura 5.20: Imagen con los controles de los mandos

- **Mover hacia delante el control**. Aparece rayo de teletransporte. Si se tiene seleccionado sobre un teletransporte y se suelta el control, el jugador se teletransporta.
- **Mover hacia atrás el control**. Si la mano que sujeta el mano está mirando en dirección a un objeto, aparece un rayo atractor. Si se lleva hacia atrás Axis2D.Thumbstck, se atraerá dicho objeto.
- **Mover hacia la izquierda o derecha el control**. Si se mueve este control ya sea hacia la izquierda o hacia la derecha, se rotará la cámara 30 grados

Respecto a los controles que se llevan a cabo mediante gestos, existen dos tipos de gestos:

- **Gestos con la cabeza**. Durante el inicio de la experiencia, si el jugador asiente con la cabeza, da a entender al robot que ha entendido la explicación que se le está dando. Si el jugador niega con la cabeza, da a entender al robot que no ha entendido la explicación que se le está dando.
- **Gestos con la mano**. Si el jugador levanta la mano a partir de cierto momento de la experiencia, dará a entender al jugador que no sabe qué hacer en ese momento, y el robot procederá a ayudarle.

#### 5.1.2.7. Jugabilidad

El jugador podrá, dentro del área de movimiento establecida, moverse de forma libre. Podrá moverse con todo el cuerpo, especialmente utilizando las manos y la cabeza.

#### Mecánica

Las posibles mecánicas que puede realizar el jugador se dividen en dos categorías: las que necesitan los mandos para realizarse y las que necesitan únicamente los movimientos del jugador para llevarse a cabo.

- Mecánicas realizadas con los mandos:
  - **Coger un objeto**
  - **Lanzar un objeto**
  - **Soltar un objeto**
  - **Usar un objeto**
  - **Atraer un objeto**
  - **Pegar un puñetazo**
  - **Tocar al robot**
  - **Señalar a alguien o algo**
  - **Teletransportarse**
  - **Levantar la mano**
  - **Mover la cámara**
- Mecánicas realizadas con el movimiento del jugador:
  - **Pulsar un botón**
  - **Agacharse**
  - **Mirar a alguien**
  - **Gestos**:
    - Asentir
    - Negar
  - **Acercarse mucho al robot**:
    - Por delante
    - Por detrás

#### Dinámica

El juego no tiene un estado de perder ni uno de ganar como tal. Lo que implicaría ganar en el juego sería lograr salir de la casa, que es el objetivo principal e implica haber completado correctamente todas las tareas que hay a lo largo de la experiencia.

No dispone de un sistema de puntos como tal, pero sí que se puede entender e interpretar cuándo se ha hecho algo bien mediante la actitud que

adopta el robot en ese momento. Si el jugador hace algo bien, el jugador se alegrará o te aplaudirá transmitiendo así el mensaje de que se ha hecho bien, mientras que si por ejemplo el jugador ha creado una poción incorrecta, el robot se mostrará triste.

**Estética**

La estética de la escena de ejemplo está basada en un ambiente de fantasía y magia (véase Figura 5.21), y esto se puede observar por el diseño y la decoración de la casa, y por los objetos que hay repartidos por ella como pociones, una varita, un fantasma que flota, o una hoguera con una cara sonriente.

Figura 5.21: Imagen de la estética basada en un ambiente de fantasía y magia

#### 5.1.2.8. Objetos

**Objetos interactivos** Se puede interactuar con la mayoría de objetos que hay por la escena. Vamos a enumerar cuales son:

- **Llave**. Este objeto es necesario para poder abrir el cajón y cumplir una de las primeras tareas a realizar. Se pone coger, soltar y lanzar, y al colocarla sobre la cerradura del cajón, lo abre.
- **Libro perteneciente al atril**. Este libro es necesario para poder activar los teletransportes y al abrirse contiene cierta información de cómo teletransportarse. Se puede coger, soltar y lanzar, y al colocarlo sobre el atril se abre.

- **Radio**. se puede interactuar con este objeto ya sea subiendo el volumen o eligiendo la cadena a reproducir, haciendo que se reproduzca una melodía. Se puede coger?
- **Libros**. Estos libros no tienen ningún uso como tal pero se pueden coger, soltar y lanzar.
- **Pociones pertenecientes a la sala**. Estas pociones no tienen ningún uso como tal, pero al igual que los libros se pueden coger,soltar y lanzar.
- **Fantasma**. No tiene ningún uso como tal, pero al igual que los libros se puede coger, soltar y lanzar.
- **Ingredientes**. A pesar de haber varios tipos de ingredientes, únicamente tres realmente tienen una utilidad en sí, y es que forman parte de la receta necesaria para poder crear la poción que abrirá la puerta. Estos ingredientes son: el filete, la pluma y la calabaza. Todos los ingredientes se pueden coger, soltar y lanzar.
- **Caldero**. El caldero es indispensable para poder crear la poción que abrirá la puerta. La tapa inicialmente está cerrada y se abre solo si se dispara con la varita sobre la diana que hay en la tapa. Tiene una zona donde se pueden distinguir distintos controles que modifican los parámetros de la poción: dispone de una rueda que marca la temperatura del caldero, dispone del modo de remover el líquido interior del caldero y por último tiene un botón que sirve para fabricar una poción, ya sea errónea o correcta. No se puede coger, solo interactuar con él.
- **Poción incorrecta**. Esta poción se puede obtener de varias maneras: o bien haber metido los ingredientes incorrectos o haber establecido los parámetros de la poción de manera errónea. No tiene ninguna utilidad de por sí, y se puede interactuar con ella de la misma manera que con el resto de pociones.
- **Poción correcta**. Esta poción se puede obtener al haber tenido en cuenta todas las instrucciones de la receta, es decir haber metido en el caldero los ingredientes correctos con los parámetros correctos. Cuando se vierte el contenido de la poción sobre el colador de la puerta, se provoca la apertura de la puerta.
- **Varita**. Esta varita se encuentra en una estantería y es necesaria para poder abrir el caldero. Cuando se usa, se dispara un hechizo que impacta sobre lo que tenga delante. Si impacta sobre el robot, este cambiará de color. Si se dispara a la diana de la tapa del caldero, este se abrirá. Se puede coger, soltar, lanzar y usar.

- **Carteles *in situ***. Los carteles que se encuentran en la habitación no tienen ninguna utilidad salvo la propia información que contienen. Se pueden coger, soltar y lanzar.
- **Mesas y estanterías**. Algunas mesas y estanterías de la habitación tienen cajones que pueden ser abiertos y que tienen objetos en su interior. Se puede interactuar con ellas agarrando los pomos de sus cajones y abriendo sus puertas.
- **Troncos**. Estos troncos se encuentran dentro de un baúl que está al lado de la chimenea. No tienen utilidad como tal. Si se lanza alguno de estos troncos al fuego, estos empezarán a quemarse. Se pueden coger, soltar y lanzar.
- **Fuego**. El fuego se encuentra en la chimenea. No tiene utilidad como tal. Provoca que los troncos se quemen cuando entran en contacto con él. El jugador no puede interactuar directamente con él de ninguna manera.

#### Objetos decorativos

Existen algunos objetos en la habitación con los que no se puede interactuar y son meramente decorativos.

- **Cuadros**: estos se encuentran por las paredes de la casa.
- **Muñeco en el techo** Este muñeco a pesar de parecer que puede ser interactivo, es solo decoración de la casa.
- **Resto de elementos de la habitación**. Dichos objetos, como son: ventanas, paredes, techo, farolillos, baúl, calabaza encima de la puerta, tuberías y suelo son solo elementos decorativos, no se puede interactuar con ellos.

## 5.2. Implementación

En esta sección se abarcará como se ha implementado la herramienta y la demostración, así como las tecnologías utilizadas para ello.

### 5.2.1. Tecnologías

Uno de nuestros objetivos era desarrollar esta herramienta como complemento de **Unity**, y así ha sido realizada. Unity es uno de los motores

de videojuegos más relevantes en el mundo de los videojuegos hoy día. Permite desarrollar videojuegos para múltiples plataformas, utiliza C# como lenguaje de programación y está orientado a componentes.

Hemos elegido este motor por distintos motivos. El primer motivo es, porque tal y como se ha explicado en la introducción de este trabajo, es el motor de videojuegos más utilizado en el país, y además hemos hecho uso de él múltiples veces a lo largo de la carrera, por lo que nos resulta más fácil de utilizar que cualquier otro motor. Otro motivo de gran peso es que Unity proporciona herramientas y facilidades para el desarrollo de videojuegos en RV que nos van a permitir desarrollar la herramienta de manera más rápida y eficiente. Para poder programar utilizaremos Visual Studio (Microsoft, 1997), ya que es es el editor de scripts predeterminado e integrado de lenguaje C# para Unity.

Además, han sido utilizados diferentes recursos integrados dentro de Unity:

#### Behavior Designer

Todo el comportamiento que tiene el NPC para simular la presencia social y el rol de instructor, se ha gestionado mediante un árbol de comportamiento creado en Behavior Designer.

Behavior Designer es una herramienta creada para la implementación de árboles de comportamiento. Posee un editor visual con el cual se pueden crear las tareas de manera sencilla.

La elección de esta herramienta se basa en que es uno de los recursos más populares en Unity, incluso han tenido colaboraciones directas con Unity Technologies. Además, hemos hecho uso anteriormente de ella en asignaturas como Inteligencia Artificial para Videojuegos. Debido a esto, conocemos cómo funciona y podemos trabajar de manera más rápida.

No solo se ha utilizado el módulo principal de Behavior Designer, también se ha utilizado Behavior Designer Movements (Opsive, 2014b), útil para las acciones de movimiento del NPC.

De hecho, nuestra herramienta se acoplará a Behavior Designer de la misma manera que lo hace Behavior Designer Movements: un paquete de tareas y además, árboles de comportamiento externos con comportamientos predefinidos (véase Figura 5.22).

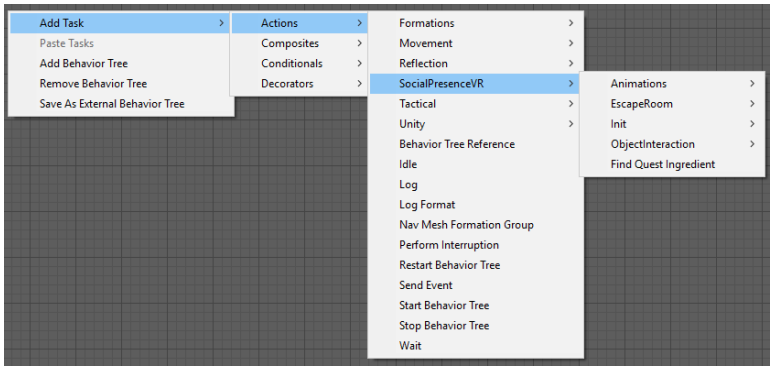



Figura 5.22: Imagen de ejemplo de creación de tarea de Social Presence VR

### VRTK

VRTK (Extend Reality Ltd, 2018). Es una herramienta que contiene una colección de facilidades para crear un entorno de RV. Su objetivo es ayudar a la productividad acelerando el proceso de creación desde prototipar ideas hasta crear soluciones completas.

El desarrollo comenzó con la utilización de este kit de desarrollo, sin embargo, acabó siendo descartado y reemplazado, debido a que se encontraba desactualizado y no se consiguió un buen funcionamiento conectándolo con las demás herramientas, especialmente las de detección de gestos.

### VR Beginner: The Escape Room

VR Beginner (Unity Technologies, 2019). Es un *asset* creado por los desarrolladores de Unity que contiene un entorno de RV y proporciona facilidades para poder realizar ciertas acciones básicas de RV. Todo esto se encuentra en una pequeña escena de ejemplo que consiste en un *Escape Room*, en el cual el usuario debe salir de la casa en la que se encuentra.

Este es el *asset* principal en el que se basa nuestra demostración de la herramienta. La escena que utilizamos es la que proporcionan, a la que le añadimos un NPC y su comportamiento.

### VRGestureRecognizer

VRGestureRecognizer (KorinVR, 2013). Es una herramienta de detección de gestos de la cabeza para RV. Permite detectar gestos como el de asentir o negar con la cabeza.

### MiVRy

MiVry (MARUI-PlugIn, 2019). Es una herramienta de detección de gestos de las manos para RV. Permite al usuario mover el mando como si se tratara de una varita, registrando así diferentes símbolos o gestos y poder asociarlos a acciones específicas.

### Dotween

Dotween (Demigiant, 2015) es un motor de animación orientado a objetos, optimizado para C#. Utilizado para realizar corrutinas de movimiento fácilmente. En nuestro caso ha sido utilizado para la creación de un efecto de levitar en el NPC y carteles.

### Graphy

Graphy (Tayx, 2018) es un monitor y depurador de estadísticas, fácil de usar y repleto de funciones para el proyecto. Su principal característica es monitorizar los fps (del inglés *frames per second*), característica necesaria para evitar detectar bajones de fps en la RV.

### Arte y música

Con respecto al arte, hemos utilizado principalmente los *assets* proporcionados por VR Beginner, y además hemos utilizado los siguientes para la integración del NPC en la escena:

- **Modelos**. Se ha utilizado el *asset* Jammo Character (Mix and Jam, 2019) como NPC para la demostración de la herramienta.
- **Animaciones**. Todas las animaciones se han extraído de la página Mixamo (Adobe Inc., 2008), que dispone de una galería de animaciones bastante amplia cuyas animaciones son compatibles con muchos modelos.
- **Partículas**. Se ha tomado como base el *asset* Unity Particle Pack (Unity Technologies, 2018) para la creación de efectos de partículas para el NPC.

Además, de forma excepcional, hemos utilizado (Adobe Systems Incorporated, 1990) para la creación y modificación de algunas texturas que hiciesen falta encajar al NPC dentro de la escena.

La música y sonido proviene de páginas de distribución con licencias libres:

- **Música**. Se ha obtenido de Patrick de Arteaga (de Arteaga, 2015).
- **Sonidos**. Se han obtenido diferentes sonidos en Freesound (Freesound team, 2015).

### 5.2.2. Herramienta

En este apartado se procederá a explicar las acciones, condiciones y conjuntos de comportamiento realizados para la herramienta.

#### 5.2.2.1. Acciones

Las acciones de la herramienta se han dividido en diferentes categorías como son las acciones ya predefinidas por Behavior Designer, las referentes a las animaciones, las referentes al sonido, al movimiento, a la interacción con objetos y a la inicialización y referencias.

##### Acciones Behavior Designer

Una parte de las acciones listadas en el apartado de diseño, se encuentran ya desarrolladas previamente por Behavior Designer (véase Figura 5.23), por lo que la implementación propia no ha sido necesaria, puesto que se puede encontrar un funcionamiento similar al establecido.

Las acciones ya provistas por Behavior Designer son las siguientes:

- **Estar quieto**. Realizada mediante la tarea *Idle*.
- **Esperar**. Realizada mediante la tarea *Wait*.
- **Efectos especiales**. Realizada mediante la tarea *Play particles*.
- **Enseñar cartel**. Esta acción es realizada mediante la conjunción de las siguientes acciones (véase Figura 5.24):
  - ***Instantiate***. Instancia un objeto.
  - ***Destroy***. Destruye un objeto en el tiempo establecido.
  - ***Set parent***. Establece el objeto como hijo de un *GameObject*.
  - ***Set local position***. Establece la posición local del objeto.
  - ***Set local rotation***. Establece la rotación local del objeto.

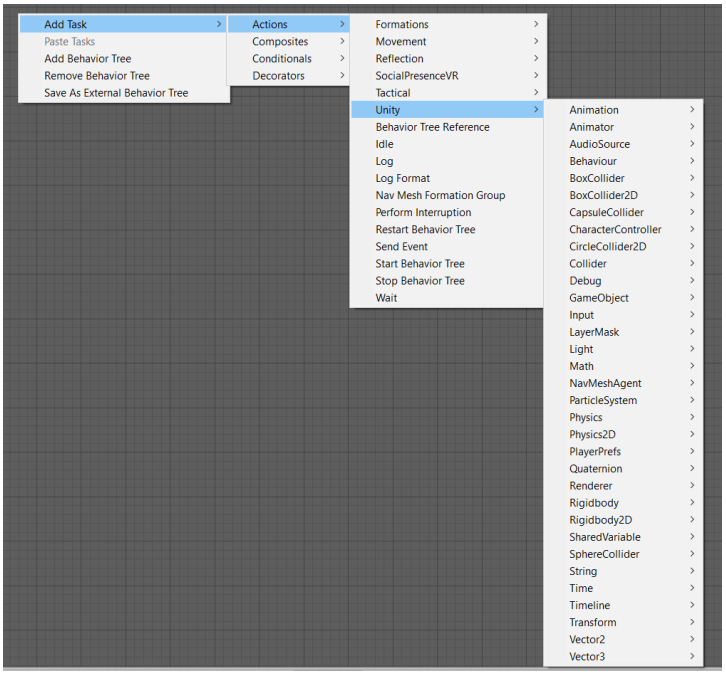



Figura 5.23: Listado de las acciones disponibles por Behavior Designer

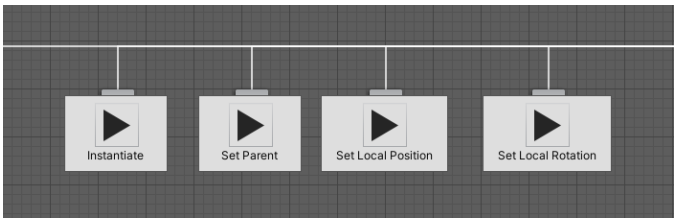



Figura 5.24: Conjunto de acciones utilizadas para realizar la acción de enseñar cartel

### Animaciones

La **simplicidad** es uno de los requisitos de nuestro diseño, y la implementación tiene que ajustarse al diseño establecido y hacer que el usuario no tenga que preocuparse de crear maquinas de estados y/o transiciones entre estas.

El manejo de animaciones dentro de árboles de comportamiento es complejo, y pueden realizarse diferentes implementaciones del mismo. La documentación de Behavior Designer recomienda dos modos (Opsive, 2014d):

- Reproducir las animaciones mediante tareas en el **árbol de comportamiento** que modifican los parámetros de la máquina de estados de

animaciones del personaje y hace que transite de un estado a otro (véase Figura 5.25).

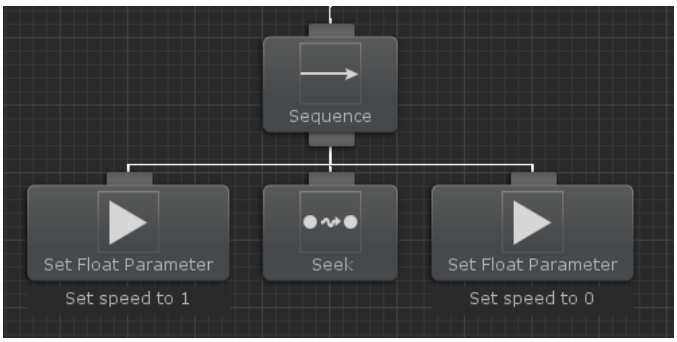



Figura 5.25: Primer modo de animar NPCs recomendado por Behavior Designer

- No reproducir las animaciones mediante tareas en el árbol de comportamiento y realizarlo en un ***script* controlador** del estado del NPC. Por ejemplo, cuando el NPC se mueve por una malla de navegación, este *script* observará la velocidad y establecerá la animación correspondiente en función de ese valor.

El **primer modo** funciona correctamente y es sencillo cuando se tienen árboles poco complejos, pero en cuanto el número de animaciones crece, este método empieza a resultar bastante incómodo y además, cuando realizas un cambio en la máquina de estados de animaciones, tienes que rehacer el árbol de comportamiento aplicando el nuevo cambio.

Por otro lado, el **segundo modo** es el indicado para árboles complejos, como es el caso de los capaces de desarrollar con esta herramienta, pues permite no preocuparse de las animaciones dentro del árbol de comportamiento. Sin embargo, hemos concluido que esta aproximación no es la adecuada para nuestra herramienta, puesto que no tener que preocuparse de crear nodos de animación dentro del árbol de comportamiento hace que el usuario tenga que preocuparse de realizarlo en *scripts* externos cuando quiera crear una nueva animación, y este no es el resultado que queremos.

Este segundo modo funcionaría perfectamente si esa preocupación fuese solo tarea de nosotros, los desarrolladores. Sin embargo, nuestra herramienta está pensada para crear NPCs que proporcionen una respuesta constante a todas las acciones del jugador, y siendo la mayoría de estas respuestas animaciones, es fundamental que el usuario pueda reproducir las animaciones a su gusto y no solo para los casos que nosotros definamos.

Por lo tanto, el sistema de animaciones propuesto se basa en el primer modo y tratará de fortalecer sus dos debilidades:

- Simplificar al máximo la creación de nodos de animaciones para reducir la complejidad del árbol.
- Evitar que el usuario tenga que complicarse rehaciendo el árbol del comportamiento cuando añada o modifique animaciones.

Para que el usuario no tenga que complicarse modificando la máquina de estados, los estados, transiciones y parámetros serán creados en ejecución. Convirtiendo esto en una complejidad para los desarrolladores de la herramienta, pero no para los usuarios que la utilicen.

La implementación en nuestra herramienta se ha realizado de la siguiente manera:

1. En la acción **Inicializar Social Presence NPC** se crea una maquina de estados nueva, con una transición al estado inicial y se asocia al componente *Animator* del agente.
2. Posteriormente, las tareas de reproducir animación crean al inicio de la ejecución, esto es, en el *OnAwake* de las tareas de Behavior Designer, un nuevo estado, al que se le asocia la animación correspondiente, el tiempo de transición, la transición al mismo y un *trigger* que activa esa transición. Esta transición se realiza desde el estado *AnyState*, es decir, desde cualquier otro estado, se puede llegar al creado si el *trigger* se ha activado (véase Figura 5.26).

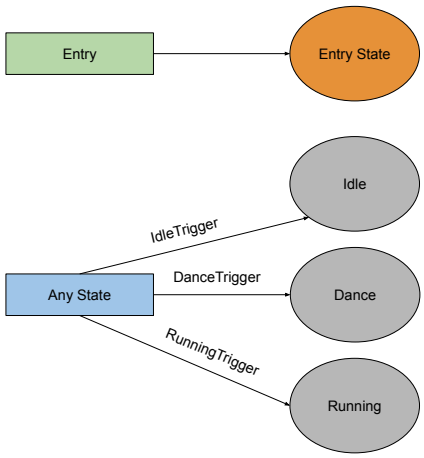



Figura 5.26: Representación de la maquina de estados de la animación del NPC

3. Por si otro nodo utiliza la misma animación, se guarda el nuevo estado creado en una lista, la cuál siempre es consultada antes de realizar el paso anterior para comprobar si ya existe ese estado.

4. A la hora de tener que empezar a realizar la acción, se activará el *trigger* asociado. Dependiendo de si es el nodo con bucle o sin él, la tarea parará de ejecutarse cuando haya finalizado la animación o cuando haya sido interrumpida.

Además, para tener la posibilidad de pasar clips de animación, ha sido necesario crear una nueva *SharedVariable*, esto es, variables que pueden aparecer dentro del inspector de la herramienta de BehaviorDesigner, para que puedan establecerse fácilmente las referencias a la animación correspondiente (véase Figura 5.27).

```
[System.Serializable]
4 referencias
public class SharedAnimationClip : SharedVariable<AnimationClip>
{
    public static implicit operator SharedAnimationClip(AnimationClip value) { return new SharedAnimationClip { Value = value }; }
}
```

Figura 5.27: *SharedVariable* de *AnimationClip*

### Comunicación sonora

Behavior Designer proporciona una de las dos acciones propuestas en el diseño, esta es **Reproducir sonido**.

El problema de esta tarea, es que si se quiere realizar un comportamiento que incluye reproducir un sonido en bucle y este comportamiento puede ser interrumpido, cuando sea interrumpido, este sonido no parará de reproducirse y tiene que ser parado manualmente (véase Figura 5.28).

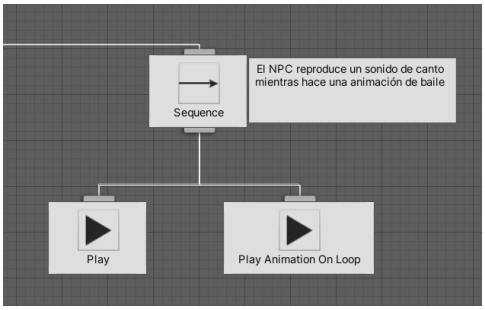



Figura 5.28: Ejemplo de comportamiento de reproducción de sonido que puede ser interrumpido

Debido a ello, se ha diseñado e implementado la tarea de **Reproducir sonido en bucle**, la cual, cuando es interrumpida, parará de reproducir la fuente de audio.

### Movimiento y rotación

La implementación de las tareas de movimiento y rotación del agente ha venido dada principalmente por la utilización del paquete de Behavior Designer Movements (véase Figura 5.29). Este ha proporcionado la solución para las siguientes tareas:

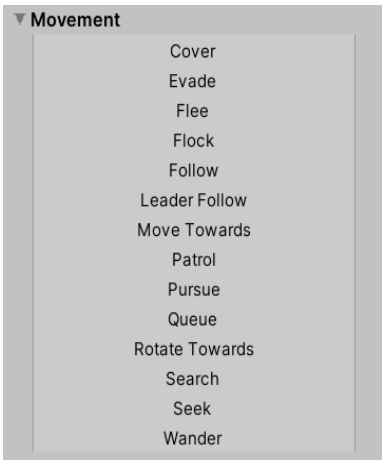



Figura 5.29: Acciones que proporciona el paquete Behavior Designer Movements

- **Desplazarse hacia un punto**. Implementada como *Seek*.
- **Seguir al objetivo**. Implementada como *Follow*.
- **Evadir al objetivo**. Implementada como *Evade*.
- **Rotación hacia un punto**. Implementada como *Rotate towards*.

Además, Behavior Designer por defecto, proporciona la implementación de **Teletransportar un agente a un punto** mediante la tarea *Warp*.

El desarrollo de la implementación de **Dar la espalda al jugador** fue realizada en los inicios del proyecto, sin embargo, nos encontramos con muchos problemas por la posibilidad de que esta tarea fuese interrumpida por otra. Debido a un largo intento de implementación y por la necesidad de tener que avanzar en otros aspectos, se acabó por prescindir de esta tarea hasta un desarrollo futuro.

#### Interacción con objetos

La implementación de la acción de **Recoger objeto** realiza lo siguiente:

1. Detecta si puede recoger el objeto, en cuyo caso prosigue.
2. Guarda la información del objeto en el componente auxiliar *SP_NPC*.
3. Ancla la posición del objeto a la mano del NPC.

**Entregar objeto** está implementado de la siguiente manera:

1. Detecta si puede entregar el objeto, en cuyo caso prosigue.
2. Se suscribe a los eventos del objeto, guardado en *SP_NPC*, de ser agarrado por el jugador.
3. Cuando el objeto es agarrado, se desuscribe de los eventos de ese objeto y lo desancla de la mano del NPC.

#### Inicialización y referencias

La acción de **Inicializar Social Presence NPC** se encarga de hacer que todas las demás tareas funcionen correctamente.

Añade al NPC el componente *SP_NPC*, en el cual se guardan todas las referencias que utilizarán las diferentes tareas. Por ejemplo la altura del jugador o la referencia a qué objeto está agarrando el NPC actualmente.

Además, si la variable *booleana* **Resetear animaciones** se encuentra activada, se encargará de añadir el componente *Animator* y crear la máquina de estados con su estado inicial. Esto hará que las diferentes acciones de animación, creen un nuevo estado y una transición hacia el mismo. Esto solo es posible en el editor, pues Unity no permite crear o modificar maquinas de estados en el ejecutable.

En cuanto a las tareas de referencias, son tareas muy simples que devuelven la referencia correspondiente (manos o cabeza del jugador) y detectan error en caso de no encontrarla.

### 5.2.2.2. Condiciones

En esta sección se tratará la implementación de las condiciones realizadas para la herramienta.

### Condiciones corporales

La tarea de **Ver al jugador** es implementada por Behavior Designer Movements en la tarea *Can See Object*, sin embargo, el funcionamiento que este ofrece no es el deseado para nuestra herramienta.

*Can See Object* utiliza un campo de visión en 3D, es decir, crea un cono de generatriz **Distancia de la vista** a partir de la posición del NPC. Esto resulta en comportamientos indeseados al detectar distancias muy cortas. Por ejemplo, en el caso de la detección de la mano del jugador muy cerca, situando el vértice del cono en los ojos del NPC, si las manos se encuentran muy cerca de la parte inferior del cuerpo del NPC, estas no se encuentran dentro de su cono de visión, y por tanto, devolverá que no las está viendo.

Para evitar este comportamiento indeseado, se ha tomado como referencia la implementación de **Can See Object** de Behavior Designer y creado una nueva condición, la cuál, en vez de detectar el objeto en un cono de visión, lo detecta en toda la proyección en la coordenada Y del triángulo de visión (véase Figura 5.30).

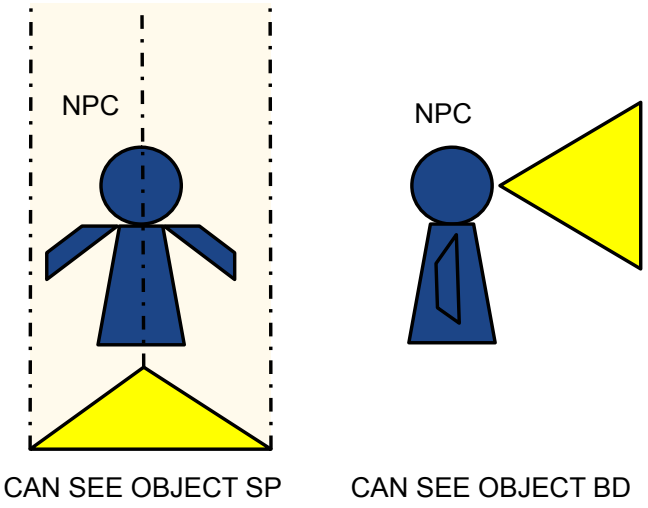



Figura 5.30: Diferencias en el campo de visión entre *CanSeeObject* de Behavior Designer y el implementado

La implementación de **Jugador mira fijamente a un objeto** difiere de la de **Jugador mira al NPC**. La primera se encarga de comprobar una mirada fija y directa hacia un objeto durante un tiempo mientras que la segunda se encarga de comprobar si el jugador tiene en su campo de visión al NPC.

**Jugador mira fijamente a un objeto** lanza un rayo de colisión desde cada uno de los ojos para detectar si choca con el objeto establecido, y si mantiene la mirada fija durante los segundos establecidos sin dejar de colisionar los rayos en ningún instante, la condición se cumplirá.

En cuanto a **Jugador mira al NPC**, la implementación se basa en la comparación de los ángulos entre el campo de visión y el formado entre la orientación del jugador y el vector de distancia entre NPC y jugador (véase Figura 5.31).

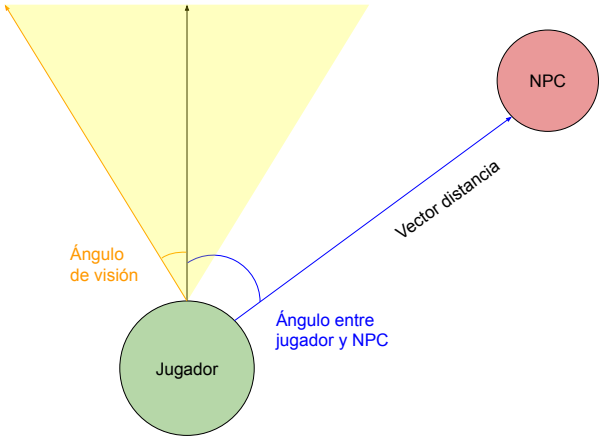



Figura 5.31: Representación de la implementación de *Jugador mira al NPC*

La condición **Jugador quieto** se ha desarrollado de la siguiente manera:

1. Se ha creado un componente auxiliar *PlayerStillState* encargado de observar durante el juego si el jugador permanece una cantidad de tiempo quieto. Esto se ha realizado en otro componente y no dentro de la propia condición para evitar que si la condición es utilizada múltiples veces en el mismo árbol, se estuviese comprobando si está quieto en diferentes lados a la vez.

2. Este componente auxiliar dispone de dos parámetros *Threshold* y *CheckRate*. *Threshold* es la máxima cantidad de movimiento en la que se asume que el jugador está quieto y *CheckRate* cada cuanto tiempo se comprueba si el jugador está quieto.

3. En el método *Update* se comprueba si han finalizado los *CheckRate* segundos y si la cantidad de movimiento es menor que *Threshold* (véase Figura 5.32).

4. La condición de **Jugador quieto** se encarga de consultar a *PlayerStillState* una *booleana* que indica si el jugador está quieto.

```
//Se obtiene la nueva posición del jugador
actualTargetPos = new Vector3(targetObject.position.x, targetObject.position.y, targetObject.position.z);

//Se halla la diferencia de movimiento entre la posición actual y la anterior
offsetPos = actualTargetPos - lastTargetPos;

//Acumulacion de la variacion de movimiento a lo largo del tiempo
offsetSum += offsetPos.magnitude;

//Se comprueba si la diferencia de movimiento supera el threshold de movimiento establecido
positionChanged = offsetSum > Threshold;

//Se intercambian valores. La posición anterior ahora es la actual
lastTargetPos = new Vector3(actualTargetPos.x, actualTargetPos.y, actualTargetPos.z);
```

Figura 5.32: Secuencia de código de acumulación del *Threshold* en el componente *PlayerStillState*

La implementación de **Jugador está agachado** es muy simple. En función del parámetro de **Porcentaje de agachado** se calcula la altura a partir de la cual se considera que el jugador se ha agachado y este valor se puede comprobar con la altura actual del jugador para detectar si se cumple la condición.

### Interacción del jugador con el entorno

La implementación de la detección de la interacción del jugador con el entorno, requiere un previo conocimiento acerca de como funciona esta interacción con objetos en los componentes previstos por Unity.

El componente ***XRGrabInteractable*** representa a aquellos objetos que el jugador puede agarrar, usar, atraer y lanzar. Este componente dispone de eventos que son lanzados en momentos concretos de la interacción y se permite la suscripción desde cualquiera parte a ellos (véase Figura 5.33).

```
public XRInteractableEvent onHoverExit { get; set; }
public XRInteractableEvent onFirstHoverEnter { get; set; }
public XRInteractableEvent onHoverEnter { get; set; }
public XRInteractableEvent onLastHoverExit { get; set; }
public XRInteractableEvent onSelectEnter { get; set; }
public XRInteractableEvent onSelectExit { get; set; }
public XRInteractableEvent onActivate { get; set; }
public XRInteractableEvent onDeactivate { get; set; }
```

Figura 5.33: Eventos disponibles en los objetos XRGrabInteractable

De estos eventos, los más relevantes para la interacción son:

- **OnFirstHoverEnter**. Lanzado cuando el jugador apunta al objeto con el laser atractor.

- **OnLastHoverExit**. Lanzado cuando el jugador deja de apuntar al objeto con el laser atractor.
- **OnSelectEnter**. Lanzado cuando el jugador agarra el objeto.
- **OnSelectExit**. Lanzado cuando el jugador suelta el objeto.
- **OnActivate**. Lanzado cuando el jugador usa el objeto.
- **OnDeactivate**. Lanzado cuando el jugador deja de usar el objeto.

Las condiciones de **Jugador usa un objeto** y **Jugador señala un objeto** se han implementado de la siguiente manera:

1. Se ha creado una condición abstracta *IsPlayerInteracting* de la cual heredarán ambas condiciones. Dispone de los atributos comunes y de unos métodos abstractos que deben ser implementados, en los cuales deben observar o dejar de observar los eventos del objeto correspondiente (véase Figura 5.34).

```
/// <summary>
/// Método que tiene que ser suscrito.
/// Añade el evento de empezar interacción correspondiente
/// </summary>
3 referencias
protected abstract void AddOnListener();

/// <summary>
/// Método que tiene que ser suscrito.
/// Añade el evento de acabar interacción correspondiente
/// </summary>
3 referencias
protected abstract void AddOffListener();

/// <summary>
/// Método que tiene que ser suscrito.
/// Deja de escuchar al interactable actual
/// </summary>
4 referencias
protected abstract void RemoveListeners();
```

Figura 5.34: Métodos abstractos en *IsPlayerInteracting*

2. Además, esta clase se encarga de gestionar cuando un objeto nuevo es recibido, para desuscribirse de los eventos del objeto anterior y empezar a estar atento al nuevo.
3. Cuando se produzca el evento correspondiente, una *booleana* se actualizará y en base a ella, la condición será cumplida o no.

4. La implementación estas condiciones (véase Figura 5.35) se basa en la utilización de los eventos de *XRGrabInteractable*.

```
protected override void AddOnListener()
{
    interactable.onActivate.AddListener(On);
}
```

Figura 5.35: Implementación del evento de empezar interacción de *Jugador usa un objeto*

La condición **Jugador coge un objeto** se ha implementado otra solución que no requiere de este sistema de observadores. Esto es gracias a que el componente *XRGrabInteractable* dispone de una booleana indicadora de si el objeto está siendo agarrado o no y basta con consultar esa variable en la condición.

Con respecto a la interacción con teletransportes, estos objetos tienen otro componente derivado desde la clase padre de *XRGrabInteractable*, y por tanto, mantienen los mismos eventos, solo que son lanzados en momentos diferentes. El más relevante para la detección del teletransporte es ***OnSelectExit***, el cuál se lanza cuando el jugador se teletransporta al nodo de teletransporte. Para la implementación de **Jugador se teletransporta**, la condición se suscribe a este evento de todos los nodos de la escena para así retornar *Success* cuando se produzca.

### Interacción física del jugador con el NPC

Behavior Designer proporciona la condición **HasEnteredCollision**, la cual devuelve *Success* cuando se produce una colisión con el NPC. A partir de esta tarea pueden ser creadas las acciones diseñadas (véase Figura 5.36).

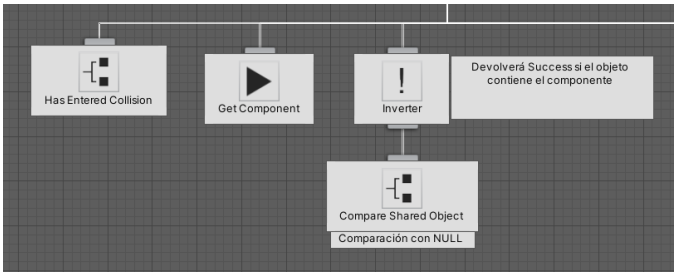



Figura 5.36: Estructura de la detección de colisión con el objeto correspondiente

Siguiendo esta estructura, se ha conseguido implementar la condición de **Ser golpeado por un objeto**. A pesar de que esta estructura también parece útil para detectar **Ser tocado por el jugador**, no ha sido posible su correcto funcionamiento. Esto parece deberse a que la detección de colisiones con objetos que hacen el seguimiento de los controladores RV no funciona de manera precisa.

Debido a esto, la implementación de la detección de ser tocado por el jugador con la cabeza o con las manos, ha sido realizado con la condición **Ver al jugador** explicada anteriormente, utilizando un valor del radio de detección similar a a la cápsula de colisiones del NPC.

La tarea de **Ser golpeada por el jugador** no ha sido implementada debido a falta de tiempo de desarrollo. La implementación debería tratar de detectar la colisión y la velocidad a la que se está desplazando la mano, para considerar si se trata de un golpe o simplemente de un toque.

### Gestos del jugador con la cabeza

La detección de estos gestos ha sido posible gracias a la utilización del *asset* VRGestureRecognizer, el cual permite detectar los gestos de asentir y negar con la cabeza.

Esta herramienta realiza esta detección a través del componente ***VRGestureRecognizer***, el cual dispone de eventos a los que suscribirse (véase Figura 5.37).

```
/// <summary>
/// Lanzado cuando se detecta el gesto de asentir
/// </summary>
public event Action NodHandler;

/// <summary>
/// Lanzado cuando se detecta el gesto de negar
/// </summary>
public event Action HeadshakeHandler;
```

Figura 5.37: Eventos proporcionados por *VRGestureRecognizer*

Este *script* no se encontraba bien parametrizado y los valores de detección se encontraban escritos directamente en código. Ha sido necesaria una modificación para parametrizar estos atributos y así, poder encontrar los valores acordes para una correcta detección de los gestos (véase Figura 5.38).

```
/// <summary>
/// Cada cuantos segundos se detecta si se ha realizado algún gesto
/// </summary>
[SerializeField] float recognitionInterval = 0.5f;

/// <summary>
/// Rango de detección del gesto de asentir
/// </summary>
[Header("Nod")]
[SerializeField] float minNodValue = 5f;
[SerializeField] float maxNodValue = 10f;

/// <summary>
/// Rango de detección del gesto de negar
/// </summary>
[Header("Headshake")]
[SerializeField] float minHeadShakeValue = 5f;
[SerializeField] float maxHeadShakeValue = 10f;
```

Figura 5.38: Parámetros de la detección de gestos con la cabeza

Para la implementación de esta detección dentro de una condición, se ha optado por tener un *script* auxiliar ***HeadGestureState*** encargado de suscribirse a los eventos del componente anterior y almacenar si se ha realizado el gesto en un valor *booleano* de fácil acceso para las condiciones. Cuando se produce un gesto, su *booleana* devolverá cierto hasta que haya pasado una cantidad de tiempo establecida desde el editor.

Se ha optado por realizar así ya que dentro de un mismo árbol pueden existir multitud de nodos de detección de gestos con la cabeza, y si se suscribiese cada nodo, el coste de ejecución aumentaría considerablemente, en vez de simplemente consultar el valor de una *booleana*.

### Gestos del jugador con las manos

La implementación de la detección de los gestos con las manos comenzó con la utilización de la herramienta MiVRy. Esta permitía la detección de gestos con un movimiento continuo, como por ejemplo saludar o levantar la mano.

Esta herramienta permite la creación de tus propios gestos, mediante la grabación de la realización del gesto, creando una base de datos. La detección de los gestos se realiza mediante una red neuronal a partir de los atributos grabados: posición y rotación de la cabeza y de las manos.

Se realizó una implementación utilizando esta herramienta parametrizando todos los valores referentes a la detección de gestos, para que fuese altamente personalizable (véase Figura 5.39).

```
/// <summary>
/// Similitud entre en el gesto de la base de datos y el detectado para que sea detectado como gesto realizado
/// </summary>
[Range(0f, 1f)]
public float GestureSimilarity;

/// <summary>
/// Intervalo en segundos en el que se realiza la identificación del gesto.
/// </summary>
[SerializeField] private float RecognitionInterval = 0.1f;

/// <summary>
/// Duración aproximada en segundos de los gestos a detectar.
/// </summary>
[SerializeField] private float GesturePeriod = 1.0f;

/// <summary>
/// Archivo del que se cargan los gestos.
/// </summary>
[SerializeField] private string LoadGesturesFile = "Sample_Continuous_Gestures.dat";
```

Figura 5.39: Parámetros de la detección de gestos con las manos

Los resultados no fueron buenos, la red neuronal era muy poco precisa y a pesar de hacer multitud de pruebas modificando parámetros y grabando repetidas veces los gestos, no fue posible un funcionamiento correcto. Esto supuso una gran perdida de tiempo de trabajo y como consecuencia, tuvimos que descartar la detección de gestos complejos como **Saludar**. A partir de entonces, decidimos que solo detectaríamos gestos simples que no requiriesen de utilizar ninguna herramienta y fueran fácilmente implementados totalmente por nosotros.

Previamente a ello, observamos que para la detección correcta de los gestos simples, era necesario conocer como estaban dispuestos los dedos de las manos. Por ello, se creó un *script* controlador de los mandos denominado ***ControllerState***. El cual está asociado a cada uno de los mandos y se encarga de observar el input y almacenar el estado actual de las manos (véase Figura 5.40).

```
/// <summary>
/// Enum con los posibles estados de animación de la mano
/// </summary>
12 referencias
public enum HandState { OPEN, CLOSE, POINTING };
```

Figura 5.40: Diferentes estados de animación de la mano

La implementación del gesto de **Señalar** consiste en detectar que el estado de la mano se encuentra en *Pointing* y comprobar mediante un rayo de colisión lanzado desde el dedo índice que colisiona con el objeto establecido a señalar.

La condición del gesto de **Levantar la mano** detecta si solo una de las manos está levantada, considerando que está levantada si la diferencia entre cabeza y mano es mayor que la establecida como parámetro, y además comprueba si la mano se encuentra en el estado de *Pointing* o *Open*.

#### 5.2.2.3. Conjuntos de comportamientos

Siguiendo el diseño establecido, se han creado conjuntos de comportamiento que son árboles de comportamiento externos dentro de Behavior Designer (véase Figura 5.41).

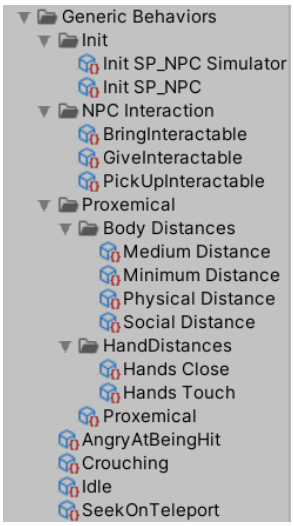



Figura 5.41: Árboles de comportamiento externo genéricos creados para la herramienta

Cada uno de estos árboles dispone de parámetros de entrada necesarios para poder funcionar. Por ejemplo el árbol *AngryAtBeingHit* necesitará la animación y el sonido a reproducir y una referencia al jugador para saber su posición.

A continuación, se va a mostrar la implementación de algunos conjuntos de comportamientos que forman parte de la herramienta, que han sido representados mediante árboles de comportamiento de Behavior Designer que nos han resultado más interesantes de mostrar:

- Buscar jugador y entregar objeto.

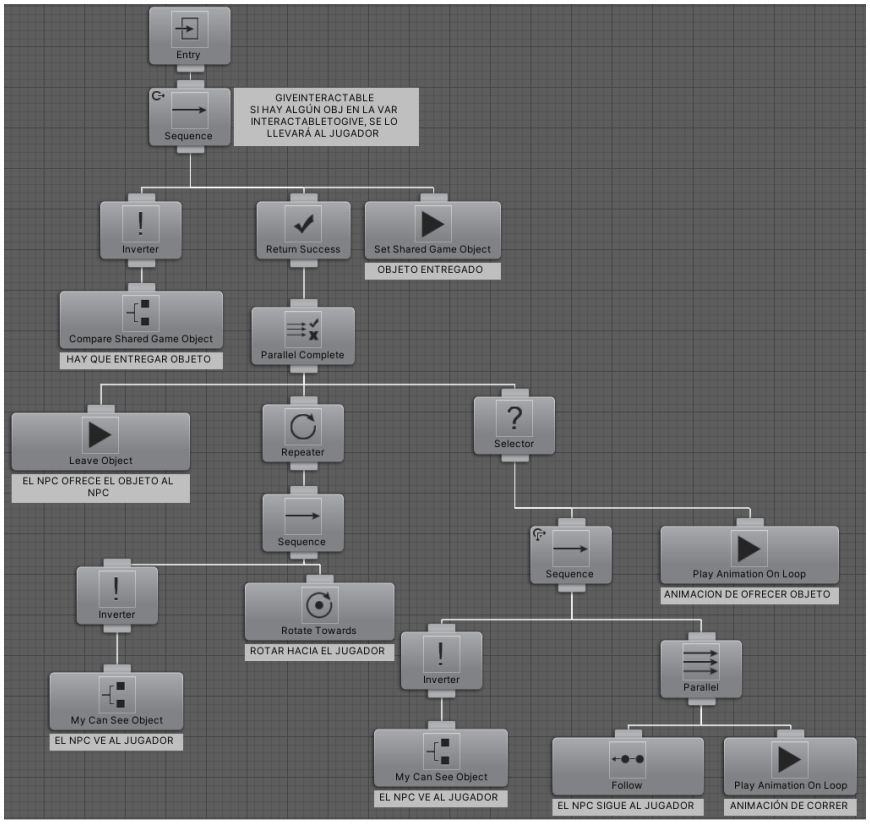



Figura 5.42: Árbol de comportamiento de buscar jugador y entregar objeto

- Explicar cartel.

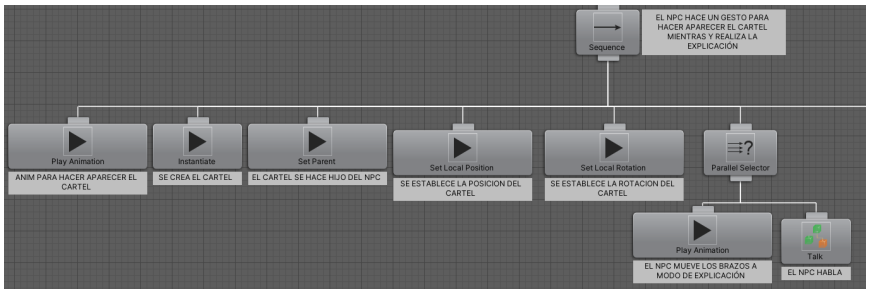



Figura 5.43: Árbol de comportamiento de explicar cartel

- Comportamiento proxémico.

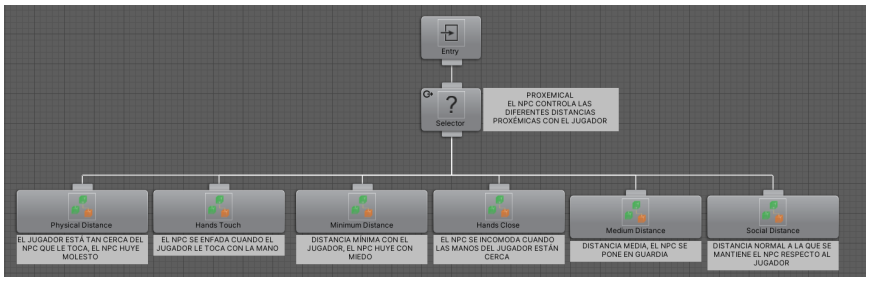



Figura 5.44: Árbol de comportamiento de comportamiento proxémico

- Distancia social con el jugador.

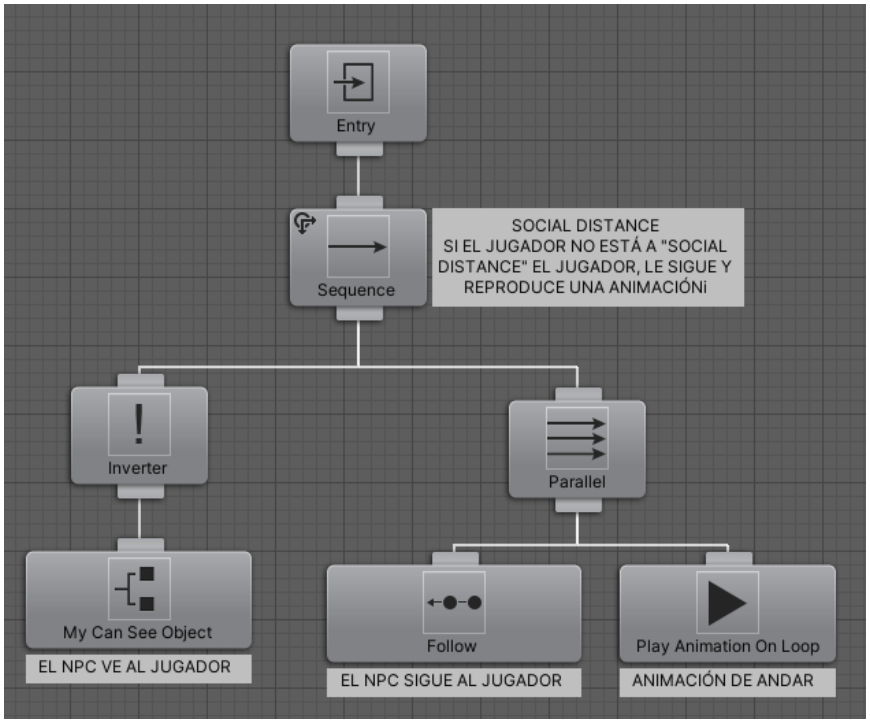



Figura 5.45: Árbol de comportamiento de distancia social con el jugador

### 5.2.3. Demostración

En este apartado se procederá a explicar las acciones, condiciones y conjuntos de comportamiento realizados para la escena de ejemplo.

#### 5.2.3.1. Acciones

Las acciones de la herramienta se han dividido en dos categorías que son las acciones que afectan físicamente a Jammo y las referentes a las pruebas.

**Acciones que afectan físicamente a Jammo**

Existen dos posibles acciones que afectan directamente al cuerpo de Jammo, ya a sus ojos y su forma, o al color de parte de su cuerpo.

- **Cambiar los ojos de Jammo**. Esta acción modifica el material de los ojos del robot (véanse Figuras 5.46 y 5.47). Jammo puede tener cuatro emociones: normal, enfadado, contento y K.O. Cada emoción está relacionada con un número respectivamente. Para poder modificar los ojos, se le pasa el número a un método de un *JammoEyesController*, que dispone de toda la información referente a los ojos y sus posibles materiales.

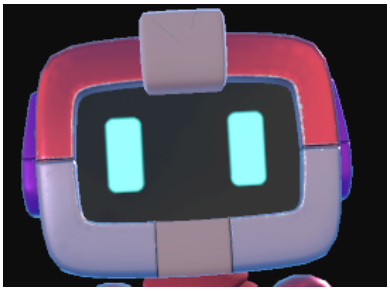


Figura 5.46: Imagen de Jammo en estado normal.

Figura 5.47: Imagen de Jammo en estado feliz.

- **Cambiar el material de Jammo**. Encargada de modificar la textura del robot. Tiene 4 posibles colores: rojo, azul, negro y dorado.

**Acciones relacionadas con las pruebas**

- **Encontrar ingrediente de la prueba**. El robot comprueba qué ingredientes necesarios para la receta se encuentran en el área establecida, y elige uno de ellos. Además, comprueba si el ingrediente encontrado no se encuentra ya en la lista de ingredientes que tiene el caldero, ya que según la receta, solo se puede meter una copia de cada ingrediente correcto.

#### 5.2.3.2. Condiciones

Las condiciones de la escena de ejemplo se han dividido en dos categorías, las condiciones relacionadas con las pruebas y las referentes a la interacción con objetos del entorno.

#### 5.2.3.3. Condiciones de fin de fase

Estas condiciones de comprobación de fin de fase funcionan todas de forma similar. Cada una de ellas se suscribe a un evento correspondiente al objeto que da paso a la siguiente fase. Estas condiciones son las siguientes:

- **Libro abierto**. Cuando ocurre este evento, se pasa a la Fase 2.
- **Caldero abierto**. Cuando ocurre este evento, se pasa a la Fase 3.
- **Puerta abierta**. Cuando ocurre este evento, se pasa a la Fase 4.

**Condiciones relacionadas con las pruebas**

- **Primer ingrediente metido en el caldero**. El robot comprueba si el tamaño de la lista de ingredientes que tiene el caldero es superior a 0, es decir, si se ha metido un ingrediente inicial.
- **Poción elaborada**. Esta acción se suscribe al *OnBrew* (véase Figura 5.48) que posee el componente *CauldronContent*. Si los ingredientes que se encuentran en la lista de objetos del caldero corresponden a los de la receta, entonces devolverá que la poción creada es correcta, y en caso contrario, devolverá que la poción creada es incorrecta.

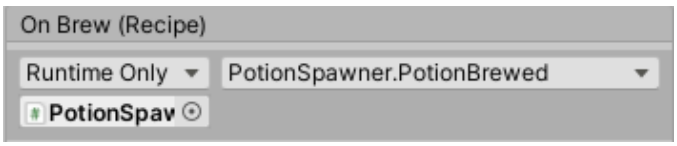



Figura 5.48: Método del caldero al que se suscribe nuestro método *poción elaborada*

**Condiciones referentes a la interacción con objetos del entorno**

- **Radio encendida** Accede al componente *Radio*, y comprueba si algún audio de la radio se está reproduciendo y si el volumen es mayor o igual al que el que se ha pasado como parámetro. Este componente de la radio posee información de las posibles canciones que tiene la

radio, del control de cuándo se cambia de canción y de cuándo se ha modificado volumen.

#### 5.2.3.4. Conjuntos de comportamientos

A continuación, se va a mostrar la implementación de algunos conjuntos de comportamientos que forman parte de la escena de ejemplo, que han sido representados mediante árboles de comportamiento de Behavior Designer que nos han resultado más interesantes de mostrar:

- El robot ayuda al jugador después de colocar el primer ingrediente en el caldero.

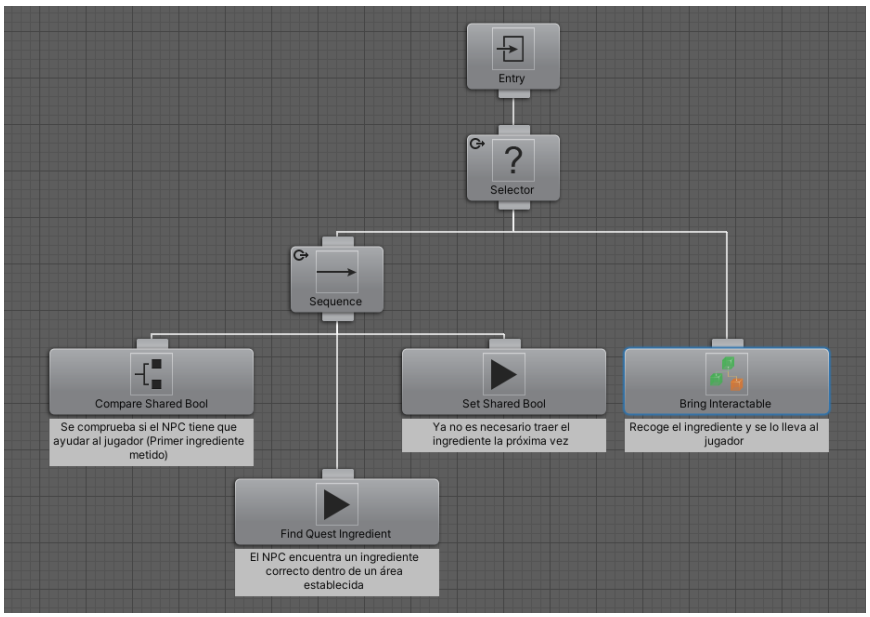



Figura 5.49: Árbol de comportamiento ayudar al jugador después de colocar el primer ingrediente en el caldero.

- Explicación en caso de que el jugador levante la mano

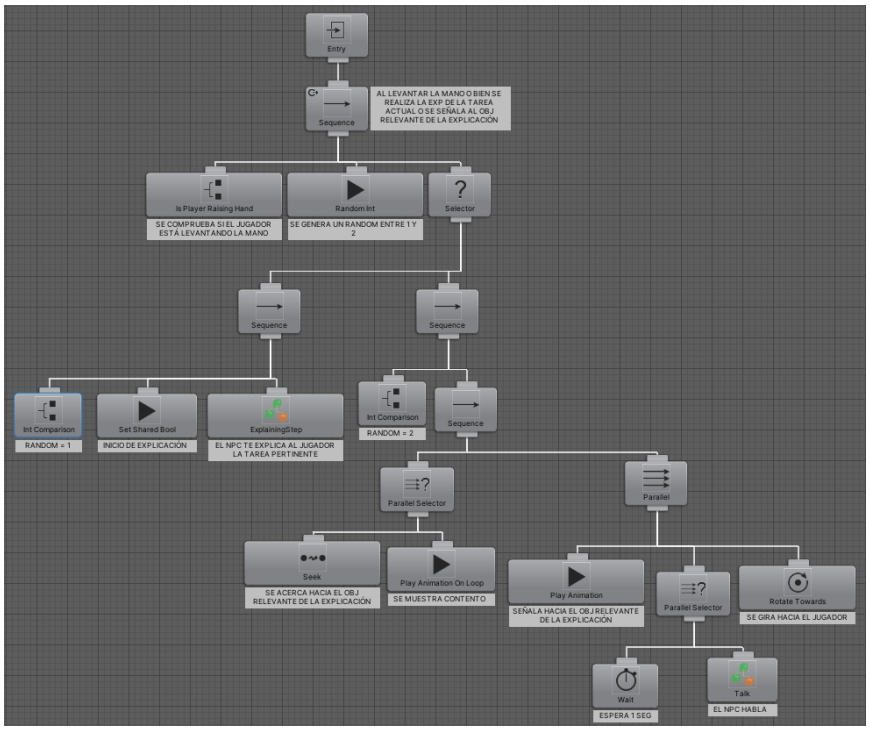



Figura 5.50: Árbol de comportamiento de explicación al levantar la mano

- Explicaciones iniciales

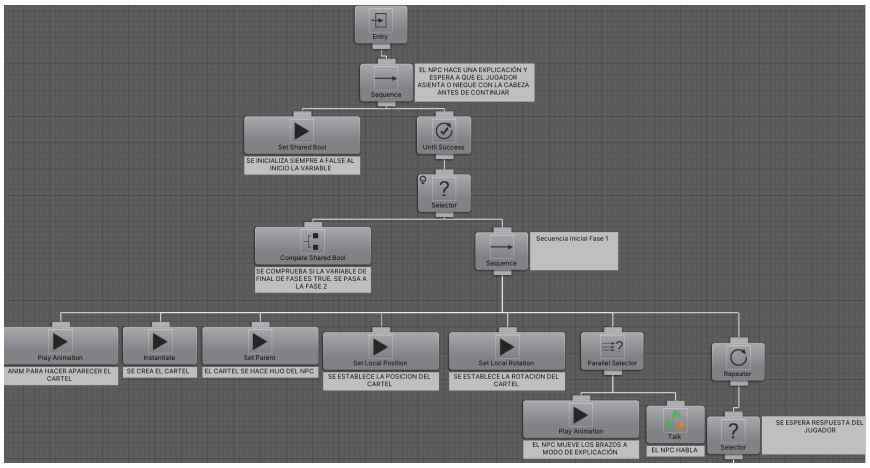



Figura 5.51: Árbol de comportamiento de explicación inicial: Parte 1

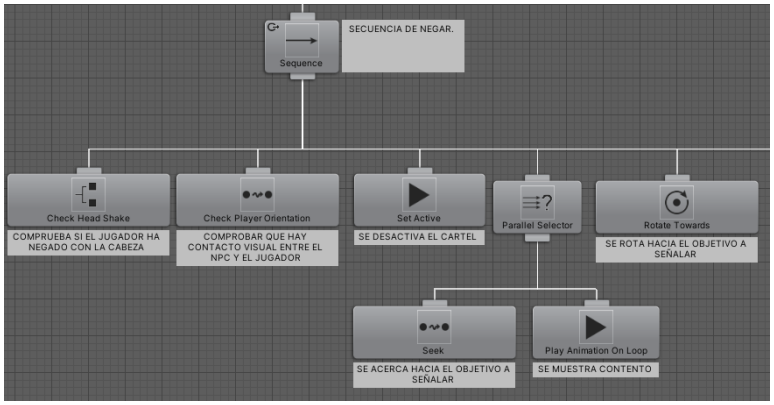



Figura 5.52: Árbol de comportamiento de explicación inicial: Parte 2

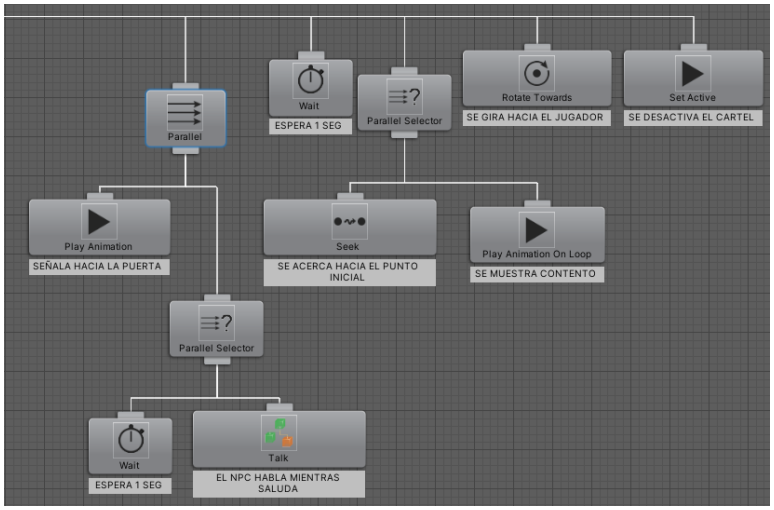



Figura 5.53: Árbol de comportamiento de explicación inicial: Parte 3

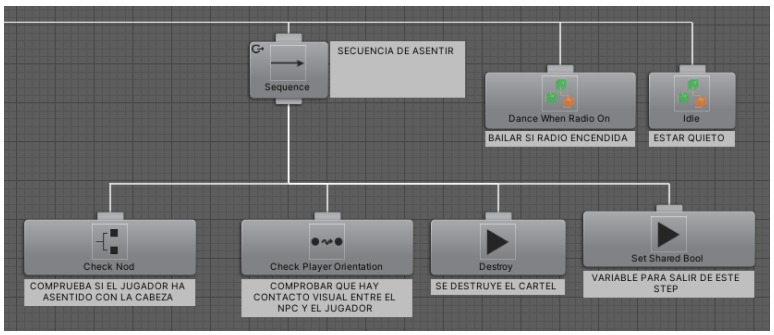



Figura 5.54: Árbol de comportamiento de explicación inicial: Parte 4

Existen algunos comportamientos que no están implementados como árboles externos, pero quedan como tarea pendiente de realizar que comentaremos al final, en la sección de trabajo futuro.

Capítulo 6

# Pruebas, resultados y discusión

En este capítulo trataremos cómo hemos desarrollado las pruebas, los resultados de las mismas y discutiremos acerca de los resultados de este proyecto.

## 6.1. Pruebas

En el Capítulo 3 se especificó que realizaríamos **test A/B** con usuarios para comprobar tanto la eficacia de la herramienta, como la capacidad que tienen los NPCs interactivos de mejorar la sensación de presencia social del jugador.

El desarrollo de la prueba se ha efectuado tal y como se especificó inicialmente, de manera que un total de 8 usuarios reales se han dividido en dos categorías de 4 usuarios cada una, según cuál de las dos versiones diferentes de la demo ha probado:

- **Usuario A**. Jugará a la demo con un NPC genérico que imita el comportamiento común en videojuegos convencionales. No usa ninguna función específica construida por nuestra herramienta. Explicará al jugador inicialmente cual es su objetivo y a continuación únicamente le seguirá hasta la finalización del juego.
- **Usuario B**. Jugará a la demo con un NPC interactivo, que reacciona a una gran cantidad de acciones del jugador. Usa todas las funciones construidas por la herramienta y además añade algunas propias de la escena. Se encarga de hacer un seguimiento activo del jugador, ayudándole cuando se encuentre perdido.

Para garantizar la coherencia en el experimento, y que los usuarios de un tipo no tengan ventaja sobre los del otro por disponer de más información, toda la información que el jugador puede recibir del NPC en el test B (véase Figura 6.2), puede ser encontrada mediante carteles disponibles a lo largo de la escena en el test A (véase Figura 6.1).

Figura 6.1: Información en test A

Figura 6.2: Información en test B

Al principio estaba previsto realizar las pruebas con usuarios de manera presencial, utilizando nuestros propios dispositivos RV y haciendo un seguimiento presencial de todo lo que realizase o comentara el jugador. Sin embargo, debido a la crisis ocasionada por la pandemia de COVID-19, las pruebas no han podido realizarse de manera presencial. Debido a ello, se optó por distribuir las demos y un cuestionario online para ambos grupos de usuarios.

Para animar a potenciales probadores a que realicen la prueba, hemos elaborado un cuestionario no muy extenso intentando evitar la pérdida de interés y asegurarnos de que aquellos que la realicen la completen sin dificultades. Asimismo, hemos publicitado e incentivado con el sorteo de un videojuego el evento de pruebas a través del sitio web y las redes sociales de Narratech Laboratories.

Se especificó la utilización de algún formulario estándar: bien SUS o bien TPI. Debido a la posible pérdida de interés que podría ocasionar los testers, se descartó el uso del formulario TPI, debido a que es muy extenso. Por otra parte, el formulario SUS dispone únicamente de 6 preguntas, por lo que se ha integrado fácilmente dentro del repertorio de cuestiones a valorar de nuestro experimento.

A parte de utilizar un formulario estándar, también se han elaborado preguntas específicas de nuestro proyecto para explorar las sensaciones dentro de la experiencia concreta. El formulario se ha organizado de la siguiente manera:

1. Introducción, requisitos y explicación de cómo realizar el experimento.
2. Preguntas previas a la realización de la prueba para conocer al tipo de usuario y su opinión acerca de la RV.
3. Preguntas posteriores a la realización de la prueba para conocer sus sensaciones en la demo y que acciones ha realizado.
4. Preguntas referentes a la sensación de presencia. Son todas las preguntas del formulario SUS.
5. Opiniones y sugerencias.

El formulario completo puede encontrarse en el Apéndice E.

## 6.2. Resultados

Toda la información detallada acerca de los resultados de las pruebas realizadas se encuentra en el Apéndice F. A continuación se van a exponer los resultados de forma general.

En general, hemos podido observar que los usuarios que han realizado la prueba tienen una edad situada entre los 20 y los 30 años y que el dispositivo más utilizado es el modelo de Oculus Rift S, precisamente el mismo que hemos utilizado nosotros durante el desarrollo. Además por los informes recibidos de los probadores, la demo ha funcionado en otros dispositivos de RV sin haber podido probar nosotros su correcto funcionamiento en ellos, dando a entender que la herramienta consigue su objetivo de ser verdaderamente multiplataforma.

Los usuarios que han realizado la prueba suelen jugar semanalmente a videojuegos de RV, por lo que podria decirse que están experimentados con esta tecnología. Además, la mayoría de los usuarios están muy de acuerdo en el hecho de sentir que están presentes dentro del mundo virtual y en que los juegos de RV hacen buen uso de la interacción con el entorno virtual y objetos.

Donde hay más disparidad de opiniones es en el hecho de si los juegos de RV hacen buen uso de la interacción con NPCs, en el cual hay gente tanto de acuerdo como en desacuerdo. Creemos que esto apunta a que al menos un sector de los jugadores de RV ven carencias en este sentido.

La gran mayoría ha podido completar el juego con éxito, y en cuanto a la duración, se ha hallado la mediana de los tiempos de cada grupo, obteniendo que la mediana del grupo A ha sido 13,25 minutos mientras que la mediana del grupo B ha sido 11,5 minutos. A pesar de no existir una diferencia muy grande, se puede observar que los usuarios que han realizado la prueba con nuestra herramienta integrada han tardado menos tiempo.

A continuación, se muestra una valoración acerca de las preguntas relacionadas con el comportamiento del NPC durante la prueba. Los datos mostrados en la tabla son referentes a los porcentajes de las valoraciones *Muy de acuerdo* y *De acuerdo*:

| Grupo | **A** | **B** |
|---|---|---|
| El robot ha ayudado al jug. a jugar de manera más eficiente | 0 % | 25 % |
| El robot se comporta de manera inteligente | 0 % | 50 % |
| El jug. ha sentido que el robot estaba pendiente de él | 50 % | 75 % |
| El jug. ha sentido que el robot reaccionaba a sus acciones | 25 % | 75 % |
| El jug. ha sentido al robot como un verdadero compañero | 25 % | 25 % |

A pesar de que la población no sea lo suficientemente grande como para poder obtener unos mejores datos, se puede observar que, en general, las valoraciones realizadas por el grupo B respectivas a los comportamientos del NPC que implican una presencia social, son superiores a las del grupo A. Por tanto, podemos interpretar que el NPC en el cual se ha integrado nuestra herramienta resulta más interactivo y dotado de una presencia social mayor.

En cuanto a los momentos en los cuales el jugador se ha podido sentir perdido durante la prueba parecen corresponderse con causas referentes al propio escenario de ejemplo de Unity que utilizamos como base, en lugar de causas referentes al NPC como tal.

Los aspectos a destacar de la prueba, tanto los positivos como los negativos, al igual que se ha mencionado en el párrafo anterior, también han sido referentes a la propia escena de ejemplo más que al NPC en sí.

Respecto a la pregunta referente a saber qué acciones concretas referentes a aspectos del entorno y del NPC han realizado los usuarios durante la prueba, se observa una intencionalidad por parte de los usuarios tanto de interactuar con la mayor parte del entorno, como de interactuar de diferentes maneras con el NPC.

Con respecto a los resultados del test SUS, se puede observar que los usuarios sienten que están en un entorno virtual más que en un lugar físico, como si fuera un lugar en el que han estado y resultando bastante similar a un sitio real en cuanto al tema de medidas, distancias, iluminación, etc. Consideramos que en estas preguntas se han visto reflejadas las opiniones de las personas con respecto a la interacción con la escena de Unity más que

con el propio NPC, y debido a ello, los resultados de ambos grupos son muy similares.

En la última parte del formulario hay un apartado en el cual los usuarios podían dejar cualquier comentario o sugerencia que quisieran. La mayor parte de comentarios plantean problemas o destacan aspectos del escenario que son referentes a la propia escena de ejemplo de Unity, como por ejemplo, el movimiento mediante teletransportes, o la interacción con objetos. Cabe destacar un comentario perteneciente al grupo A en el cual el usuario comenta que ha intentado interactuar de diversas maneras con el NPC, pero no ha obtenido respuesta, creyendo que había hecho algo mal, ya que esperaba más interacción por parte del NPC. Un ejemplo claro y manifiesto de la problemática que abordamos en este trabajo.

## 6.3. Discusión

A pesar de que han faltado por terminar de implementar algunas características mencionadas en el Capítulo 3, al no ser principalmente relevantes, no han afectado al correcto funcionamiento de la herramienta, pero quedan pendientes de hacer para aumentar aún más las funcionalidades de esta.

Aunque la escena de ejemplo esté basada en un tipo de NPC específico (en este caso instructor), en la cual se encuentran tareas dependientes de dicho rol, solo ha sido un ejemplo para mostrar la funcionalidad de la herramienta y para que los desarrolladores puedan tener una idea de cómo está estructurada.

La idea desde un inicio ha sido la de crear una herramienta expansible, y esto se puede observar, ya que la mayor parte de las tareas creadas son genéricas, haciendo que se puedan amoldar a otro tipo de roles que pueda asumir un NPC.

Con esta idea de tener presente la expansibilidad que queremos mantener como característica importante en nuestra herramienta, esta es factible de encontrarse en una posición relevante en el sector de los videojuegos de RV, ya que podría resultar útil a cualquier desarrollador de videojuegos de RV de Unity que tenga como requisito tener NPCs en su juego.

La finalidad de la herramienta es que sirva como una ayuda a la hora de poder crear un NPC interactivo, del cual no sea necesario implementar ciertas tareas básicas, porque nuestra herramienta ya las tiene implementadas. De esta forma, se espera conseguir que el tiempo que puedan invertir los desarrolladores en crear un NPC sea inferior al normal, y que la mayor parte del tiempo que invirtieran en el NPC fuera para el desarrollo de características específicas dependientes de sus propios videojuegos.

Como se dijo en el capítulo 4 hemos observado que, en general, los desarrolladores de videojuegos para RV hasta el momento no se han volcado especialmente en crear NPCs realmente interactivos, que hagan más realista la experiencia y susciten mayor presencia social. Normalmente, programan todo el comportamiento de los personajes para que reproduzca de manera bastante lineal sin que se pueda ver interrumpido por alguna acción del jugador. Nuestra herramienta propone un enfoque diferente para el desarrollo de NPCs en RV.

Capítulo 7

# Conclusiones

En este capítulo damos una conclusión al trabajo realizado, valorando si se han cumplido o no los objetivos establecidos y proporcionando una línea de trabajo para el futuro desarrollo, mejora y ampliaciones de la herramienta creada.

Estos eran los objetivos del trabajo, expuestos en el Capítulo 3:

- Identificar los factores que tienen mayor influencia en la sensación de presencia que tiene un jugador en RV.
- Desarrollo de una herramienta en Unity que permita crear NPCs cuyo comportamiento suscite presencia social y que sea altamente configurable.
- Creación de un escenario de ejemplo que contenga un NPC, cuyo rol sea el de instructor, utilizando la herramienta desarrollada.
- Comprobar la eficacia de la herramienta realizando pruebas con jugadores reales y midiendo la repercusión en la sensación de presencia.

A continuación, vamos a analizar si se han cumplido cada uno de estos objetivos:

- **Identificar los factores que tienen mayor influencia en la sensación de presencia social que tiene un jugador en RV**. En el Capítulo 2 exploramos cuales son estos factores, concluyendo en que los principales factores son:
  1. Movimiento del jugador.
  2. Interfaz de usuario.

3. Interacción con el entorno.
4. Personajes no jugables.

Al explorar los videojuegos de referencia, encontramos títulos que han logrado aprovechar estos factores y logran experiencias con una elevada sensación de presencia. Destaca el videojuego *Half Life: Alyx*, que recientemente ha sido capaz de utilizar todos estos factores para lograr un videojuego muy inmersivo, una auténtica superproducción de RV. Sin embargo, destacamos que el comportamiento de los NPCs de este juego no es tan sofisticado ni tan interactivo como el personaje con el que interactuamos dentro de un título menos conocido como es *Shinobu Project*.

Esta deficiencia nos muestra como en los videojuegos comerciales que llegan al gran público suele haber una preocupación exhaustiva por todos los factores que afectan a la presencia, pero no especialmente centrada en el comportamiento de los NPCs. Por ello decidimos abordar este proyecto y crear una herramienta que facilitase la creación de NPCs mucho más interactivos.

- **Desarrollo de una herramienta en Unity que permita crear NPCs cuyo comportamiento suscite presencia social y que sea altamente configurable**. Se ha conseguido crear esta de herramienta y de hecho, hemos cumplido la mayoría de las especificaciones propuestas en el Capítulo 3. Si bien algunas especificaciones no han sido cumplidas (mencionadas posteriormente en el apartado de trabajo futuro) y algunos aspectos pueden ser mejorados, el resultado final ha sido una herramienta completamente funcional y que demuestra ser útil para crear un NPC instructor con el que los usuarios pueden interactuar fácilmente gracias a las tareas y comportamientos desarrollados.

  La herramienta desarrollada presenta una base robusta que facilita la extensibilidad de la misma. Gracias a un diseño construido en torno a la interacción del usuario con la herramienta, este podrá añadir y crear nuevas tareas para sus propias condiciones o acciones e integrarlas como parte de la propia herramienta.

- **Creación de un escenario de ejemplo que contenga un NPC, cuyo rol sea el de instructor, utilizando la herramienta desarrollada**. En un inicio se planteó crear una escena de ejemplo propia y diseñar las pruebas a realizar por el jugador por cuenta propia. Se descartó la idea de modelar cada objeto de la escena, por lo que se acabaron utilizando recursos gratuitos hechos por terceros.

  Se encontró un escenario creado por Unity, que no solo era un escenario de una casa con objetos y decoración integrados, sino que ya

tenía implementada la dinámica de un *Escape Room* junto con aspectos relacionados con la interacción en RV. Esto nos permitió enfocarnos principalmente en el diseño del comportamiento del NPC, que realmente era la función principal del proyecto, en lugar de invertir demasiado tiempo en cuestiones secundarias.

El NPC de ejemplo fue desarrollado fácilmente gracias a la utilización de nuestra propia herramienta, la cual proporcionaba la mayoría de las tareas que necesitábamos. Aunque al igual que cualquier videojuego, todo escenario específico tiene sus propias características, y por tanto, fue necesario desarrollar algunas tareas propias, que encajaron sin problemas con la estructura general de la herramienta Social Presence VR.

- **Comprobar la eficacia de la herramienta realizando pruebas con jugadores reales y midiendo la repercusión en la sensación de presencia**. En un inicio se planteó la idea de poder realizar estas pruebas de manera presencial a un número de personas lo suficientemente relevante como para tener abundantes datos. Sin embargo, debido a la crisis sanitaria vivida a lo largo de estos meses, el alcance experimental se ha visto notoriamente reducido, dando lugar a que el número de usuarios que han realizado la prueba ha sido menor de lo deseado.

  A pesar de no tener un amplio muestreo, se ha podido observar que los resultados pertenecientes al grupo B (aquel que ha probado la demo con nuestra herramienta integrada) son mejores en materia de presencia social que los del grupo A. Esto nos da a entender que efectivamente la herramienta podría ser eficaz para el desarrollo y útil para marcar una diferencia notable frente al diseño clásico de un NPC pasivo y carente de detalles que susciten en el jugador la sensación de presencia social.

## 7.1. Trabajo futuro

Como se ha recalcado, la herramienta desarrollada cumple con los objetivos del Capítulo 3 y con la mayoría de las especificaciones planteadas en un inicio. Sin embargo, debido a algunos problemas técnicos y de restricciones de tiempo durante la fase de desarrollo, hay especificaciones no se han podido implementarse. De todas formas consideramos que la herramienta ha conseguido ser suficientemente funcional y útil con el conjunto de tareas y comportamientos desarrollados.

Social Presence VR potencialmente puede contener una enorme cantidad de acciones y condiciones, con ampliaciones en todos sus aspectos.

A continuación, se proporciona un breve listado de las líneas de trabajo futuro que aconsejamos explorar:

- Creación de las tareas no implementadas que habían sido especificadas:
  - Acción de dar la espalda al jugador.
  - Condición de jugador da la espalda al NPC.
  - Condición de ser golpeado por el jugador.
  - Crear condiciones para la detección de gestos continuos con las manos mediante la integración con alguna herramienta que tenga un funcionamiento correcto. Uno de estos gestos sería saludar.
  - Otros. Implementación de las ideas propuestas en especificación extras de detección de gestos y de interacción física con el NPC.
- Creación de más árboles externos que agrupen comportamientos. En ocasiones se utiliza la misma sucesión de tareas múltiples veces y no se ha encapsulado en un árbol externo.
- Creación de una guía de uso de la herramienta y tutoriales.
- Código de mayor calidad, especialmente una mayor documentación de todas las tareas.
- Proporcionar una interfaz para facilitar la integración en cualquier otra herramienta de árboles de comportamiento.
- Mejora del rendimiento en la escena de demostración, especialmente en el tratado de partículas, ya que a veces producen errores.
- Mayor cuidado visual de la demostración de la herramienta, especialmente para la transmisión de información mediante carteles.

# Bibliografía


Adobe Inc. Mixamo. Disponible en https://www.mixamo.com/#/.

Adobe Systems Incorporated. Adobe photoshop. Disponible en https://www.adobe.com/es/products/photoshop.html.

Atlassian. Trello. Disponible en https://trello.com/es.

Axosoft. Gitkraken. Disponible en https://www.gitkraken.com/.

Bethesda Game Studios. Fallout 4 VR. PC, 2017.

Cano Moreno, S. Elementos de la comunicación no verbal: Kinesia, proxemia y paralingüística. Disponible en https://elsilenciohablamucho.wordpress.com/2015/07/09/elementos-de-la-comunicacion-no-verbal-kinesia-proxemia-y-paralinguistica/.

Champandard, A. J. The Core Mechanics of Influence Mapping. Disponible en https://www.gamedev.net/articles/programming/artificial-intelligence/the-core-mechanics-of-influence-mapping-r2799/.

de Arteaga, P. Patrick de Arteaga. Music for Games and Multimedia. Disponible en https://patrickdearteaga.com/es/index-es/.

Demigiant. Dotween. Disponible en https://assetstore.unity.com/packages/tools/animation/dotween-hotween-v2-27676#description.

Desarrollo Español de Videojuegos (DEV). Libro Blanco del Desarrollo Español de Videojuegos 2018. Disponible en http://www.dev.org.es/es/publicaciones/libro-blanco-dev-2018.

Duch i Gavaldà, J. y Tejedor Navarro, H. Inteligencia artificial. Disponible en https://www.exabyteinformatica.com/uoc/Inteligencia_artificial/Inteligenca_artificial_ES/Inteligencia_artificial.pdf.

Enacademic.com. Chopsticks. Disponible en https://enacademic.com/dic.nsf/enwiki/2856023.

Epic Games. Unreal Engine. Disponible en https://www.unrealengine.com/en-US/.

Estudiofuture. The Summer Camp. Disponible en https://estudiofuture.com/.

Extend Reality Ltd. Virtual Reality Toolkit. Unity, Disponible en https://vrtk.io.

Facebook Technologies, LLC. Oculus Quest. Disponible en https://www.oculus.com/quest/.

Freesound team. Freesound. Disponible en https://freesound.org/.

Google. Drive. Disponible en https://www.google.com/intl/es_ALL/drive/.

Hocking, C. Disponible en https://clicknothing.typepad.com/click_nothing/2007/10/ludonarrative-d.html.

Ibañez, M. L. *Incrementar la presencia en entornos virtuales en primera persona a través de interfaces auditivas*. Tesis Doctoral, Universidad Complutense de Madrid, 2019.

Infoeducacion.es. Cómo ser profesor de autoescuela: requisitos, qué estudiar y cuánto gana. Disponible en https://infoeducacion.es/como-ser-profesor-de-autoescuela/.

Itreseller. El mercado de realidad virtual y aumentada crecerá un 78,5 % en 2020. Disponible en https://www.itreseller.es/en-cifras/2019/12/el-mercado-de-realidad-virtual-y-aumentada-crecera-un-785-en-2020.

KorinVR. VR Gesture Recognizer. Disponible en https://github.com/korinVR/VRGestureRecognizer.

Lombard, M., Ditton, T. B. y Weinstein, L. Measuring presence: The temple presence inventory. *in Proceedings of the 12th Annual International Workshop on Presence*, Disponible en http://astro.temple.edu/{~}tuc16417/papers/Lombard{_}et{_}al.pdf.

MARUI-PlugIn. Mivry. Disponible en https://assetstore.unity.com/packages/templates/systems/mivry-3d-gesture-recognition-143176#releases.

MECHATECH. What is a 3DoF vs 6DoF in VR? Disponible en https://www.mechatech.co.uk/journal/what-is-a-3dof-vs-6dof-in-vr.

MICROSOFT. Visual Studio. Disponible en https://visualstudio.microsoft.com/es/.

MICROSOFT. Github. Disponible en https://github.com/.

MILGRAM, S. y GUDEHUS, C. *Obedience to authority*. 1978.

MIX AND JAM. Jammo character. Disponible en https://assetstore.unity.com/packages/3d/characters/jammo-character-mix-and-jam-158456.

MUNDO VIRTUAL. ¿qué es la realidad virtual? Disponible en http://mundo-virtual.com/que-es-la-realidad-virtual/.

NEAT CORPORATION. Budget Cuts. PC, Disponible en http://www.neatcorporation.com/BudgetCuts/.

OCULUS. Vr Best Design Practices. Disponible en https://developer.oculus.com/design/bp-vision/.

OCULUS. Oculus First Contact. Disponible en https://www.oculus.com/experiences/rift/1217155751659625/?locale=es_ES.

OCULUS. First Steps. Disponible en https://www.oculus.com/experiences/quest/1863547050392688/?locale=es_ES.

OPSIVE. Behavior designer. Disponible en https://opsive.com/assets/behavior-designer.

OPSIVE. Behaviordesigner - movement pack. Disponible en https://assetstore.unity.com/packages/tools/ai/behavior-designer-movement-pack-16853.

OPSIVE. Documentación conditional aborts. Disponible en https://opsive.com/support/documentation/behavior-designer/conditional-aborts/.

OPSIVE. Syncinganimations. Disponible en https://opsive.com/support/documentation/behavior-designer/syncing-animations/.

OPSIVE. Behaviordesigner - formations pack. Disponible en https://assetstore.unity.com/packages/tools/ai/behavior-designer-formations-pack-28980?aid=1100lGdc&utm_source=aff#description.

OWLCHEMY LABS. Job Simulator. PC, PS4, Disponible en https://jobsimulatorgame.com/.

Owlchemy Labs. Rick and Morty Virtual Rick-Ality. PC, PS4, 2017.

Pérez Feijoo, H. M., Pérez Hernández, J. M., López González, L. y Caballero Bravo, C. Comunicación y atención al cliente. Disponible en https://www.mheducation.es/bcv/guide/capitulo/8448175743.pdf.

Rodriguez, C. y Rodriguez, D. Toma de decisiones cualitativas. Disponible en https://pt.slideshare.net/lamaluk/proceso-de-toma-de-decisiones-7239301/2.

Rodríguez Christensen, V. ¿Qué es la interfaz de usuario para videojuegos? *Women in Games*, Disponible en https://womeningameses.com/2018/05/13/que-es-la-interfaz-de-usuario-para-videojuegos/.

Slack Technologies. Slack. Disponible en https://slack.com/intl/es-es/.

Slater, M., Usoh, M. y Steed, A. Depth of presence in virtual environments. *Presence: Teleoperators and Virtual Environments*, vol. 3, 1994.

Tayx. Graphy. Disponible en https://assetstore.unity.com/packages/tools/gui/graphy-ultimate-fps-counter-stats-monitor-debugger-105778.

Unity Technologies. Unity. Disponible en https://unity.com/.

Unity Technologies. Unity particle pack. Disponible en https://assetstore.unity.com/packages/essentials/tutorial-projects/unity-particle-pack-127325#description.

Unity Technologies. VR Beginner: The Escape Room. Disponible en https://learn.unity.com/project/vr-beginner-the-escape-room.

Viva Dev. Shinobu Project. Disponible en https://shinobuproject.itch.io/game.

Ward, C. Understanding simulator sickness. Disponible en https://injury.research.chop.edu/blog/posts/understanding-simulator-sickness.

WriteLaTeX Limited. Overleaf. Disponible en https://www.overleaf.com/.

Xataka. La realidad virtual parece haber encontrado el precio, el formato y el catálogo para empezar a despegar. Disponible en https://www.xataka.com/realidad-virtual-aumentada/realidad-virtual-parece-haber-encontrado-precio-formato-catalogo-para-empezar-a-despegar.

Zimbardo, P. G. Stanford prison experiment: A simulation study of the psychology of imprisonment. 1972.

Appendix A

# Title, abstract and keywords

# Design and Development of Characters with Social Presence in Virtual Reality Video Games

**Authors**
**Adriana del Castillo Espejo-Saavedra**
**Raúl Serrano Gómez**

# Abstract

The year 2016 signified the reappearance of a wave of Virtual Reality products at which big companies as Facebook, Sony or HTC joined. This year emphasized on the creation of applications for the entertainment industry. Since then, there have been advances in technology, accessibility and gameplay that try to be exploited in the world of the commercial games.

Nowadays, emerging technologies allow reaching a higher level of immersion than in conventional video games. To take advantage of this, we should maintain the presence sensation during the whole experience. In particular, the player wants to interactuate with the game characters, he expects to percieve them as intelligent and attentive beings, who react to the actions and presence of the player, which is rare to find in the current video games. In order to create this kind of characters it is especially important the use of the Artificial Intelligence.

The objective of this project is to build a tool for Unity that facilitates the design and development of social characters though the definition of a set of basic events which need to have an answer from the character, and allowing its extensibility for every type of specific behaviors from a particular game.

The analysis and design of the proposed objectives concluded with a package of tasks and behaviors for non-playable characters called Social Presence VR. Furthermore, its effectiveness has been proven through A/B tests with users. The tool is free and can be found on the GitHub platform.

## Keywords

Video Game Development, Artificial Intelligence, Development Tool, Unity, Interactivity, Immersion, Instructor.

Appendix B

# Introduction

Virtual Reality (VR) has joined stronger than ever into videogames industry from 2016 and it has meant one of the most important changes in recent times for this type of entertainment (Mundo Virtual, 2016). Despite the growth of the market, it is true that there are different factors that prevent it from taking off and reach similar numbers to other technological products and services.

One of the principal reasons of its standstill is the cost of these products for the final user. The price of these dispositives and the need of conecting them to a computer with a powerful and updated graphic hardware, makes them reachable for only a few consumers. In any case these years it is seem that the passing of time and the resulting progresses in engineering allow this dispositives to be more economical and therefore more affordable. One of the most accessible VR model headset for the general public is *Oculus Quest* (Facebook Technologies, LLC, 2019), since it does not only have a reduced price (Xataka, 2020) but also it does not need a computer to be conected to, which allows the experiences in VR being popularized in a great way.

Specifically in Spain, in 2019 there has been a declined in the number of developed applications for VR, according to the White Book of Spanish Video Game Development (Desarrollo Español de Videojuegos (DEV), 2019). This is probably due to the complexity of its development, as there is a need of specialist in the subject and there is still a shortage of training. Furthermore, the developers need tools that make the job in this kind of proyects easier, with no need of so much specific knowledge.

At the present time, there are tools for development environment as known as Unity (Unity Technologies, 2005) that provides basic functionality for VR, for example, *VRTK* (Extend Reality Ltd, 2018). But in a market with such a big amount of products that are a simple translation of con-

ventional videogames (based in screens, keyboard and mouse or gamepad), novelties which can be differenciated and can take advantage of the features of VR are needed. It would be interesting to count on with more tools which make easier the creation of more inmersive virtual worlds.

Plus, one of the aspects that affects a lot the player's presence sensation are the Non-Player Characters (NPCs), that must seem believable, show themselves alive and react with naturally to the player's avatar.

## B.1. Virtual Reality

VR video games have the potential of recreating the presence sensation on the player, which is very difficult to get in conventional games like those that all of us used to enjoy on conventional screens, with keyboard and mouse, or with gamepad. The possibility of feeling a part of the virtual world and be able to interactuate with our own hands and with the movement of our body in the scene, makes presence the strongest point of this new technology.

It does not only produce an improvement on the immersion, but we also get a higher level of freedom and expressivity in the interaction in playing time. The player's input is not limited to the keystroke, joystick movements or mouse movements. Now this input is a continuous data flow, which provides constant information about the player's actions, of his head and of his hands.

However, this obvious capacity of improving the player's experience is a double edge weapon. Being present in the virtual world and having a higher level of action freedom makes any little discordance regarding what we expect from real world to have a bigger possibility of "throw us off" the experience. A poor designing of a VR game will not only make it boring, but will possibly make any physical discomfort on the player, such as vertigo, which is a regular sympthom of the called "simulator sickness" (Ward, 2018).

Developing VR games, therefore, implies that we have to develop knowing the limitations, both technological and regarding design, or counting on tools that have in mind these limitations. It requires a precise knowlegde of which are the main problems and which are the general guidelines to follow. The most difficult points are currently the player's avatar movement, the extradiegetic interface, the interaction with the virtual environment and specially, the interaction with the NPCs.

One of the biggest problems of the VR games is the direct inheritance of own conventional video games elements that works on conventional screens, keyboard, mouse or gamepad. These elements, which can work properly in this type of games, do not have to yield the same results in VR. There is a big

amount of ports from conventional famous games, for example, *Fallout 4 VR* (Bethesda Game Studios, 2017). It is a moving fact the exploration of the huge open worlds in this type of video games, but it is usual to find a lack of playability that prevent causing the same presence levels that could be possible if the title would have been developed specifically for this platform, like for example *Job Simulator* (Owlchemy Labs, 2016).

## B.2. Work Purpose

The difficulty that developing a VR game means, has notoriously caused a descent on the number of applications developed in the last year. Moreover, according to the data obtained in Spain, a 9% of the companies have difficulties for finding experts in VR.

The purpose of this work is contributing to the existence of developed tools that facilitate the creation of immersive experiences with characters in VR games. We want to help the development of teams that do not have experts on this technology or they can not afford to hire them. This is our potential customer profile.

Actually, development tools that help creating virtual environments for VR already exist, providing facilities for environment interaction, for the use of the user interface as well as for the management of the player's avatar movement. An example of this type of tools is VRTK.

However, tools that helps with the VR character development hardly exist on the market. For this reason, we have chosen to focus on a tool that facilitates the creation of NPCs whose reaction seems naturally in front of the player avatar's actions and make the player feel a social presence in a greater way. This tool wants to give the basic behaviors that every character in a game must have to feel part of the virtual world in VR, as well as allowing the developer to extend the system with reactions to specific events of his project.

The tool will be a plugin of the most used development environment nowadays, which is *Unity*. Globally, 73% of the development studios with less than 50 workers use Unity and specifically in Spain, the 85% of the studios use it compared to the second most used, *Unreal Engine* (Epic Games, 1998), with a 23% of use according to the White Book of Spanish Video Game Development.

The main functions we want in the tool to appear are: the detection of basic player avatar actions that every credible NPC should respond to, the construction of any kind of response to all these actions, and the customization of the system maded for detecting and acting in a desired way to the

new player's actions, always with the purpose of increasing the sensation of presence. The entire system is designed around the capacity to be easily improved and extended by future developers who want to contribute adding functionalities.

To illustrate the possibilities of the tool, once it is finished, we will create a demo game using the proposed system. Also, we will do A/B tests with real users, rating the experience and the sensation of presence of the same demo, but with the difference that one test will have an NPC builded with our tool and the other will not have it.

## B.3. Related subjects

Next, we'll make a description of the relations between this final project with the different areas of the degree, connecting it with some of the subjects studied.

The use of different Artificial Intelligence algorithms for the movement technique or for the decision-making, makes the *Artificial Intelligence* subject the most related with this work.

The design of both the tool and the example prototype, as well as the development of the *Game Design Document* (GDD), is supported by subjects like *Design of Videogames*, /textitDevelopment of Interactive Systems and *User Interfaces*.

The implementation of A/B tests and the user experience valoration is supported by the *Usability and Game Analysis* subject. The production of the project, the organization and the repository management, as well as tasks based on the *Agile Methodologies* subject. The *Game Engines* subject is related with the project because of the continuous use of Unity.

The subjects of *Sound in Videogames* and *Computer Music Technology* has provided the knowledge to handle audio editing software and digital music creation for the composition of sound effects and music for the prototype.

This Videogame Development Degree has a list of programming related subjects that, even though they have not a direct relation with the project topic, they are indispensable, because they have provided programming knowledges and the ability of designing architectures and structuring the problems. These subjects are *Basics of Programming*, *Data Structures and Algorithms*, *Video Vame Programming Technology*, *Graphic Computing*, *Algorithmic Methods in Problem Resolutions*, *Console Video Games*, and *Mobile Device Video Games*.

Moreover, all the subjects of *Project* and *Business practice*, apart from the programming development, they have provided teamwork skills and the capacity of develop an extensive project.

## B.4. Report structure

The structure of this project is composed of the following chapters:

- In the Chapter 1 we have exposed our proposal, presented the VR capacities, explained our motivation for making this project and we relate the significance of this project with subjects that we have had during this degree.
- In the Chapter 2 we make a revision of the state of the art in the most relevant fields of this project. We include a study of non verbal communication and the process of taking decisions that belongs to Artificial Intelligence, we also make an analyisis of the user experience in VR, and we finally close this chapter with a summary of the most relevant references for the development of the tool.
- In the Chapter 3 we introduce the main objectives of our project and we make an specification of all the requirements.
- In the Chapter 4 we explain the methodology used for the development of the project and which are the tools that we have used.
- In the Chapter 5 we make a detailed explanation of the analysis, design and implementation of our project, both for the tool and the example scene.
- In the Chapter 6 will begin with the explanation of the test and afterwards, we will discuss about the results that we have obtained to validate and evaluate the tool, being the result generally positive.
- In the Chapter 7 we make a final conclusion about the project, exploring whether the objectives have been met and proposing future lines of work.

Appendix C

# Conclusions

In this chapter we give a conclusion to the work done, exploring whether or not the objectives have been accomplished and we provide a line of work for the future development, improvement and extension of the tool.

These were the objectives of the work exposed in the Chapter 3:

- Identify the factors that that have most influence on the presence in VR.
- Development of a tool made with Unity that allows creating NPCs whose behavior provokes social presence and that is configurable.
- Creation of a sample scene that contains a NPC, whose role is being an instructor, using the developed tool.
- Prove the effectiveness of the tool through different types of tests with real users and rate the impact on the sensation of presence.

Next, we will check if each of the objectives has been accomplished.

- **Identify the factors that that have most influence on the presence in VR.** In Chapter 2 we explored what these factors are, concluding that the main factors are:
  1. Player movement.
  2. User interface.
  3. Interaction with the environment.
  4. Non-player characters.

When exploring reference video games, we can find examples that take advantage of this factors and achieve experiences that provide a high sense of presence. One of the videogames that stands out is *Half Life: Alyx*, which has been able to use all this factors to achieve a very inmersive videogame. However, it should be noted that the behavior of NPCs in *Half Life* are not very much either sophisticated nor reactive in comparision with the character of *Shinobu Project*.

This deficiency show us how in commercial video games which reach the general public, usually there are a exhaustive concern about all the factors which affect the presence except for the NPCs. This is why we start this project and develop a tool that help creating interactive NPCs.

- **Development of a tool made with Unity that allows creating NPCs whose behavior provokes social presence and that is configurable.** The accomplishment of the development of this tool and, in fact, we have accomplished the majority of the specifications proposed in Chapter 3. Although some specifications have not been accomplished (which are mentioned later in the Future work section) and some aspects can be improved, the final result has been a completely functional tool, that demonstrates to be useful for creating an interactive NPC easily, thanks to the tasks and behaviors developed.

  The developed tool have a solid base which facilitate the extensibility of itself. Thanks to a design built around the interaction of the user with the tool, the user will be able to add and create new tasks for his own conditions or actions and be able to introduce them in the tool.

- **Creation of a sample scene that contains a NPC, whose role is being an instructor, using the developed tool.** In the beginning, the idea was to create a sample scene right from the start. We decided to discard modelating each object of the scene so we have used free assets created by others.

  We found a scene that had been developed by Unity, which was not only a scene of a house and decoration that were integrated, but a scene that had already implemented the funcionality of an *Escape Room* and which also had aspects related with VR interaction. This allowed us to focus on the design of the NPC behavior, which is really the main functionality of the project, instead of spending time on secondary subjects.

  The NPC was developed easily thanks to the use of the tool, which provided the majority of the tasks that the NPC needed. Although, in the same manner that in every other videogame, all of them have their own features, and because of that, we needed to develop some specific

tasks related to our example scene (such the eyes changing task or the cauldron open task), which fitted perfectly with the developed tool.

- **Prove the effectiveness of the tool through different types of tests with real users and rate the impact on the sensation of presence.**

  Initially, we wanted to make these tests on a significant number of persons in order to have appropiate data. However, due to the Coronavirus crisis, the difussion of our test has been notoriously reduced, causing that the number of users who have made the test has been lower than the anticipated.

  Even though the sampling is not as large as it should be, we have been able to see that the results referring to B group (which is the one that has proven the demo that has our tool integrated) are better than the ones of the A group. This implies that our tool is in effect effective and useful and can make a notable difference opposite to a development of a passive and lacking in social presence NPC.

## C.1. Future work

The developed tool meets the objetives written in Chapter 3 and with the majority of the specifications. However, due to technical problems and a lack of development time, some of the specifications have not been achieved, but we consider that the tool with the set of developed tasks and behaviors has managed to be functional and useful.

In addition, this tool can potentially contain a huge amount of actions and conditions and can also have extensions in all its aspects.

A list of the future line of work is provided below:

- Creation of the tasks not implemented that had been specified:
  - Action of turning the NPC back on the player.
  - Condition of player turning his back on the NPC.
  - Condition of being hit by the player.
  - Conditions for detecting continuous gestures with the hands, through the integration of some tool that works correctly. One of these gestures would be waving the hand.
  - Others. Implementation of the extra proposed ideas in the specification about gesture recognition and physical interaction with the NPC.

- Creation of more external behavior trees that group behaviors. Occasionally, the same sequence of tasks is used multiple times and they have not been encapsulated in an external tree.
- Creation of an usage guide of the tool and tutorials.
- Improve the quality of the code, specially on a better documentation of all the tasks.
- Provide an interface to facilitate the integration of any other behavior tree tool.
- Improve the performance in the demo scene, specially with particle handling, because it sometimes produce errors.
- Better visual care of the demo scene, specially for the transmission of information through frames.

Apéndice D

# Aportaciones individuales de los autores

En este apéndice, expondremos las tareas más relevantes que han sido realizadas por cada uno de los integrantes. Cabe destacar que, en una parte de las tareas específicas ha habido colaboración mutua ya que ante la complejidad de la tarea, ha requerido de la ayuda del otro integrante del grupo.

A continuación, se mencionarán aquellas aportaciones más relevantes. En nuestro tablero de Trello, pueden encontrarse todas las tareas realizadas a lo largo de todo el proyecto, incluyendo el tiempo requerido y por cuál integrante ha sido realizado, todo ordenado por sprints.

## D.1. Adriana del Castillo Espejo-Saavedra

- Producción. Tareas relacionadas con la producción de este proyecto:
  - Todas las reuniones con el director y escrito de actas.
  - Preparar y participar en todas las presentaciones con el grupo de investigación y la exposición final.
  - Definir proyecto, posibles mecánicas, herramientas a utilizar y búsqueda de información.
  - Redefinir las distintas ideas del TFG que ha habido: Interacción del jugador con el entorno, respuesta a la autoridad, y orientación hacia aplicaciones en el videojuego.
  - Planificar producción. Definición de objetivos por hito y fechas. Planificar sprints de cada hito.
  - Búsqueda de videojuegos VR relevantes con buena interactividad.
  - GDD. Creación del documento de diseño de la demostración de la herramienta.
  - Creación del formulario del test A/B.
- Memoria. Tareas relacionadas con la creación de la memoria:
  - Organizar el trabajo y estructurarla.
  - Búsqueda de juegos de referencia con buena interactividad de NPCs.
  - Capítulo 2, Introducción y estructuración, y los siguientes apartados: Cominucación no verbal y rol del instructor, Toma de decisiones en los videojuegos y Videojuegos de referencia.
  - Capítulo 3, hechos los apartados de Tareas y comportamientos del NPC e Interfaz.
  - Capítulo 4 completo.
  - Traducción de los Apéndices: Apéndice A y Apéndice B.
  - Capítulo 5 Ánalisis, Diseño e Implementación de la demostración.
  - Capítulo 6, hecho el apartado Discusión.
  - Capítulo 7 completo, a excepción del apartado Trabajo Futuro.
- Herramienta. Tareas relacionadas con el desarrollo de la herramienta:
  - Creación del proyecto. Integración de: VRTK, Behavior Designer y modelo del NPC. Repositorio en GitHub conectado con GitKraken.

- Creación de un primer prototipo con VRTK. Árbol de comportamiento con movimiento y animaciones. El NPC huye cuando el jugador se acerca.
- Diseño de la herramienta. Condiciones y acciones. Estructuración en árboles externos.
- Búsqueda de múltiples animaciones en Mixamo.
- Sistema de animaciones en Behavior Designer: Loop y no Loop.
- Condición de jugador mira al NPC.
- Creación de los árboles de proxémica.
- Creación de la condición y del árbol de detección de cuándo el jugador se ha agachado.
- Creación de la condición y del árbol de detección del jugador dando la espalda. Descartarlo por su funcionamiento incorrecto.
- Programación de simulador de VR sin gafas conectadas y hacer que funcione con el árbol del NPC.
- Diseño de la escena del Escape Room, reacciones del NPC ante las acciones del jugador y estructuración en fases del NPC.
- Estructuración del árbol de comportamiento del NPC.
- Reelección de los assets de VR en Unity. VRTK está desactualizado. Empezar a utilizar Unity VR.
- Creación de las acciones y árboles de recoger objeto y ofrecer objeto al jugador.
- Creación del árbol de comprobación de ayudar al jugador cuando levanta la mano.
- Creación de la condición y árbol de comprobación de si el jugador está quieto.
- Búsqueda y creación de partículas de aura del robot.
- Creación de condición y árbol de explicación de carteles.
- Creación de carteles extra necesarios para ayudar en la explicación del NPC y adaptación de los originales a la escena con NPC.
- Desarrollo del árbol de comportamiento de fase 1: Saludo y explicaciones iniciales.
- Desarrollo del árbol de comportamiento de fase 3: Comportamientos generales, explicaciones y ayudar al jugador.
- Introducción del asset DotWeen y creación del movimiento que simula que el robot flota.
- Creación de un contador para el control del tiempo jugado. Se guarda en un fichero.

## D.2. Raúl Serrano Gómez

- Producción. Tareas relacionadas con la producción de este proyecto:
  - Todas las reuniones con el director.
  - Preparar y participar en todas las presentaciones con el grupo de investigación y la exposición final.
  - Definir proyecto, posibles mecánicas y herramientas a utilizar.
  - Redefinir las distintas ideas del TFG que ha habido: Interacción del jugador con el entorno, respuesta a la autoridad, y orientación hacia aplicaciones en el videojuego.
  - Planificar producción. Definición de objetivos por hito y fechas. Planificar sprints de cada hito.
  - Búsqueda de videojuegos VR relevantes con buena interactividad.
- Memoria. Tareas relacionadas con la creación de la memoria:
  - Organizar el trabajo y estructurarla. Organización de Overleaf.
  - Portada y Resumen.
  - Capítulo 1.
  - Capítulo 2. Sección de Experiencia de usuario en Realidad Virtual.
  - Capítulo 3. Introducción, Objetivos y Especificación de Limitaciones, Demostración de la herramienta y Test A/B.
  - Capítulo 5. Análisis, Diseño e Implementación de la Herramienta. Implementación de las Tecnologías.
  - Capítulo 6. Sección de Pruebas y Resultados.
  - Capítulo 7. Sección Trabajo Futuro.
  - Traducción del Apéndice C.
  - Apéndices E. F y G
- Herramienta. Tareas relacionadas con el desarrollo de la herramienta:
  - Creación del proyecto. Integración de los assets: VRTK, Behavior Designer y modelo del NPC. Creación de repositorio en GitHub conectado con GitKraken.
  - Diseño de la herramienta. Condiciones y acciones. Estructuración en árboles externos.
  - Sistema de animaciones en Behavior Designer: Loop y no Loop.
  - Diseño de la escena del Escape Room, reacciones del NPC ante las acciones del jugador y estructuración en fases del NPC.

- Estructuración del árbol de comportamiento del NPC.
- Reelección de los assets de VR en Unity. VRTK está desactualizado. Empezar a utilizar Unity VR.
- Investigar funcionamiento y hacer tutorial Unity VR.
- Condiciones del NPC de detectar las interacciones del jugador con objetos de la escena: coger, usar, señalar y teletransporte.
- Crear árbol externo y acciones de inicialización de Social Presence NPC.
- Búsqueda de assets de reconocimiento de gestos.
- Introducir asset de reconocimiento de gestos asentir y negar. Creación de las condiciones.
- Introducir asset de reconocimiento de gestos con las manos y crear condición de detección continua de gestos continuos con las manos. Acabar descartando asset.
- Condición de gesto de mano levantada.
- Condición de gesto de señalar objeto.
- Creación de la condición y árbol de comprobación de si el NPC ha sido golpeado por un objeto lanzado por el jugador.
- Arreglar condición de ver al jugador. No funciona correctamente la programada por Behavior Designer.
- Creación de la condición y árbol de comprobación de distancias proxémicas de las manos e integrarlo en árbol proxémico.
- Creación de las condiciones y acciones propias de la escena de ejemplo: Cambiar color de ojos, cambiar material, detectar música de radio y detectar cada cambio de fase.
- Desarrollo del árbol de comportamiento de fase 2: Comportamientos generales y explicaciones.
- Desarrollo del árbol de comportamiento de fase 4: Secuencia final de despedida.
- Contador de FPS para Debug dentro de la RV.
- Encontrar e insertar partículas de teletransporte del robot.
- Crear estructura y mantener orden en las carpetas del proyecto: Scripts y árboles de comportamiento externos.
- Árbol del NPC de asustarse cuando el jugador usa la varita.
- Árbol del NPC de cambiar de material cuando es golpeado por la magia de la varita.
- Árbol del NPC de cambiar de activar carteles cuando el jugador está cerca del caldero.

Apéndice E

# Cuestionario utilizado en el test A/B

A continuación se presenta el formulario creado para la realización de las pruebas. El aquí expuesto corresponde al grupo A del test A/B. El formulario del grupo B es exactamente el mismo excepto por el enlace a la descarga de la demostración.

# Experimentos sobre Sensación de Presencia Social en Videojuegos de Realidad Virtual - Instrucciones y cuestionario del grupo A

Hola a todos,

Somos Raúl Serrano Gómez y Adriana del Castillo Espejo-Saavedra, estudiantes de la Facultad de Informática de la Universidad Complutense de Madrid. Con la supervisión de nuestro director, Federico Peinado (Narratech Laboratories), hemos creado una herramienta de desarrollo de videojuegos como Trabajo de Fin de Grado y necesitamos tu colaboración para reunir datos que nos permitan comprobar su eficacia, además de detectar posibles errores que puedan ocurrir durante su ejecución.

Para realizar este experimento es IMPRESCINDIBLE disponer de un DISPOSITIVO DE REALIDAD VIRTUAL para PC, disponer de CONTROLADORES PARA AMBAS MANOS y utilizar el sistema operativo Windows.

Los videojuegos de realidad virtual permiten experimentar ciertas sensaciones al jugador con más facilidad que otros que disfrutamos habitualmente en pantallas convencionales, con teclado, ratón o controlador de juegos. La posibilidad de sentirse parte del mundo virtual y ser capaz de interactuar con nuestras propias manos y nuestro propio cuerpo, hace que la "presencia" sea uno de los puntos fuertes de esta nueva tecnología. La presencia es un fenómeno psicológico medible y se refiere a la capacidad de una experiencia virtual de provocar al usuario la sensación visceral subjetiva de "estar ahí realmente".

Precisamente para medir esto necesitamos que los jugadores habituales de realidad virtual, que disponen de medios y conocen su funcionamiento, prueben un juego creado con nuestra herramienta, un pequeño "Escape Room" de unos 10 minutos de duración, y DESPUÉS rellenen este cuestionario, de 20 preguntas cortas sobre la experiencia vivida.

Enlace a las instrucciones y el cuestionario:
https://forms.gle/W3szJKzG6dHxBszH6

Completando el cuestionario nos estás ayudando a mejorar nuestra herramienta y poder comprobar su utilidad real. Si tienes alguna duda del funcionamiento del juego o de cualquier otro aspecto del proyecto, no dudes en contactar con nosotros mediante cualquiera de los siguientes correos:
raulse01@ucm.es y adridelc@ucm.es

Muchas gracias por vuestra colaboración,



Raúl y Adriana
*Obligatorio

## Consideraciones previas a la realización de la prueba

La aplicación que vas a probar ha sido diseñada para dispositivos de realidad virtual de PC con sistema operativo Microsoft Windows. Los desarrolladores hemos utilizado un visor Oculus Rift S, aunque nuestra herramienta pretende ser multiplataforma y debería ser posible ejecutar esa aplicación sin muchas dificultades en dispositivos similares (HTC Vive, Valve Index, Oculus Rift, etc).

Es necesario disponer de un equipo informático que permita ejecutar de forma fluida videojuegos de realidad virtual. Si tu equipo no es lo suficientemente potente, es posible que la experiencia de usuario se vea alterada y no se puedan obtener los resultados deseados.

## Realización de la prueba

Descarga la APLICACIÓN a probar desde este enlace:
https://drive.google.com/open?id=1Gnf5Pqis1HqhQU1mncCFE6FpbAxyCsk3

Descomprime el contenido del fichero ZIP.

Para la correcta ejecución del fichero EXE que encontrarás en la carpeta, al tratarse de un aplicación aún en desarrollo y todavía sin verificar, es necesario que des permiso a tu dispositivo de realidad virtual para poder "ejecutar aplicaciones de orígenes desconocidos". En el caso de Oculus está opción se encuentra en la aplicación principal de Oculus, dentro de Configuración > General > Activar orígenes desconocidos.

Ejecuta el juego y tras habituarte al entorno virtual en el que te encuentras, averigua lo que debes hacer para salir de la habitación donde te encuentras y hazlo.

Aviso: Si encuentras problemas de funcionamiento dentro del juego, puedes pulsar la tecla ESCAPE para abandonarlo y volver a ejecutarlo de nuevo.

## Preguntas previas

A continuación aparecen una serie de preguntas preliminares que podrían ser respondidas sin necesidad de haber realizado la prueba.

1. 1. ¿Qué edad tienes? *

_____

2. 2. ¿Qué dispositivo de realidad virtual tienes? *

*Selecciona todos los que correspondan.*

- [ ] Oculus Rift S
- [ ] Oculus Rift
- [ ] HTC Vive
- [ ] Valve Index

Otro: [ ] _____

3. 3. ¿Con qué frecuencia sueles jugar a videojuegos de realidad virtual? *

*Marca solo un óvalo.*

- Diariamente
- Semanalmente
- Mensualmente
- Muy infrecuentemente

4. 4. En los videojuegos de realidad virtual que he jugado/juego suelo sentir que estoy presente en ese mundo. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

5. 5. Considero que los juegos de realidad virtual hacen buen uso de la interacción con el entorno virtual y sus objetos. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

6. 6. Considero que los juegos de realidad virtual hacen buen uso de la interacción con los personajes no jugadores.

*Marca solo un óvalo.*

- ◯ Totalmente de acuerdo
- ◯ De acuerdo
- ◯ Ni de acuerdo ni en desacuerdo
- ◯ En desacuerdo
- ◯ Totalmente en desacuerdo

## Preguntas posteriores a la realización de la prueba

A continuación, aparecen una serie de preguntas que deben ser respondidas DESPUÉS de haber probado nuestra aplicación, ya que son referentes a esta.

7. 1. Me ha gustado el juego. *

*Marca solo un óvalo.*

- ◯ Totalmente de acuerdo
- ◯ De acuerdo
- ◯ Ni de acuerdo ni en desacuerdo
- ◯ En desacuerdo
- ◯ Totalmente en desacuerdo

8. 2. He conseguido completar el juego (salir de la habitación). *

*Marca solo un óvalo.*

- ◯ Sí
- ◯ No

9. 3. En caso de haber completado el juego, ¿cuánto tiempo te ha llevado? Introduce los minutos que ha durado la ejecución, dato que puedes leer en el fichero TiempoJugado.txt que encontrarás en la siguiente ruta de tu ordenador: Equipo > Documentos *

10. 4. El robot me ha ayudado a jugar de manera más eficiente y rápida. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

11. 5. El robot se comportaba de manera inteligente. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

12. 6. He sentido que el robot estaba pendiente de mí y de lo que iba haciendo. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

13. 7. He sentido que el robot reaccionaba a mis acciones. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

14. 8. He sentido al robot como un verdadero compañero. *

*Marca solo un óvalo.*

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo

15. 9. ¿Ha habido algún momento concreto en el que te hayas sentido perdido? Si es así, cuéntanos cual. *

16. 10. ¿Ha habido algún momento que te haya llamado especialmente la atención? Si es así, explícanos cual. *

17. 11. ¿Cuáles de las siguientes acciones has realizado durante tu sesión de juego? *

*Selecciona todos los que correspondan.*

- [ ] Encender la radio
- [ ] Lanzar un objeto al robot
- [ ] Acercarme al robot
- [ ] Disparar con la varita al robot
- [ ] Agacharme
- [ ] Levantar la mano

## Preguntas referentes a la sensación de presencia

A continuación, aparecen preguntas referentes al cuestionario estándar Slater-Usoh-Steed (SUS), que se centra en cuestiones generales sobre la sensación de presencia y la experiencia vivida en el entorno virtual.

18. 1. Por favor, puntúa tu sensación de "estar dentro" del entorno virtual, en una escala del 1 al 7, donde 7 representa tu sensación natural de encontrarte en un lugar físico real. *

*Marca solo un óvalo.*

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

19. 2. ¿Hasta qué punto ha habido ocasiones durante la experiencia en las que el entorno virtual era la realidad para ti? *

_____

20. 3. Cuando ahora piensas de nuevo en la experiencia, ¿piensas en el entorno virtual más como una serie de imágenes que has visto o más bien como un sitio en el que has estado? *

_____

21. 4. Durante la experiencia, ¿qué sensación ha sido más fuerte, tu sensación de encontrarte en un entorno virtual, o de encontrarte en otro lugar? *

22. 5. Haz memoria sobre el momento en el que te has encontrado en el entorno virtual. ¿Cómo de similar ha sido esta experiencia a la "estructura del recuerdo" de otros lugares físicos en los que has estado hoy? *

Por "estructura del recuerdo" entendemos cosas como la medida en la cual tienes recuerdo visual del entorno virtual, si ese recuerdo es en color, la medida en la cual el recuerdo resulta vívido o realista, su tamaño, la localización en tu imaginación, la medida en la cual es panorámico en tu imaginación, y otro tipo de elementos estructurales.

23. 6. Durante la experiencia, ¿has pensado de manera frecuente que te encontrabas realmente en el entorno virtual? *

## ¡Gracias por completar el formulario!

Para terminar, hemos añadido un par de apartados en los cuales puedes dejarnos tu correo si quieres conocer los resultados de nuestro proyecto, además de un apartado libre donde puedes poner cualquier comentario que quieras para ayudarnos a mejorar nuestra herramienta.

Al finalizar, no olvides pulsar en el botón de abajo para enviar tus respuestas.

24. 1. Dirección de correo electrónico (opcional, para recibir los resultados del proyecto y poder participar en el sorteo)

Consulta las condiciones del sorteo en http://narratech.com/

25. 2. En este apartado puedes hacer cualquier comentario, sugerencia o aclaración que quieras. Tu opinión nos ayudará a seguir mejorando.

Apéndice F

# Resultados del test A/B

A continuación se presentan los resultados del formulario creado para la realización de las pruebas.

# Resultados del Test A/B

**Respuestas Test A → Rojo.**

**Respuestas Test B → Azul.**

## Preguntas previas

**Las respuestas a cada una de las preguntas previas, al ser generales, estarán agrupadas en una sola gráfica.**

1. ¿Qué edad tienes?

**23**

**24**

**21**

**28**

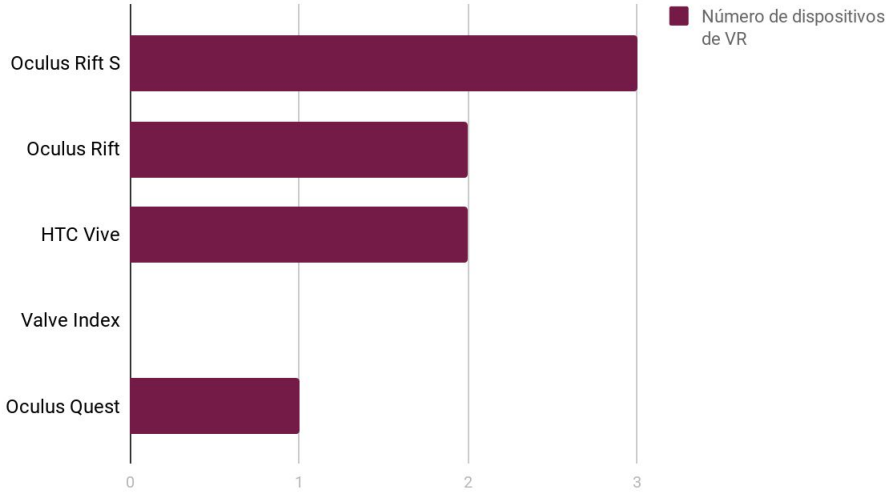
**28**

**34**

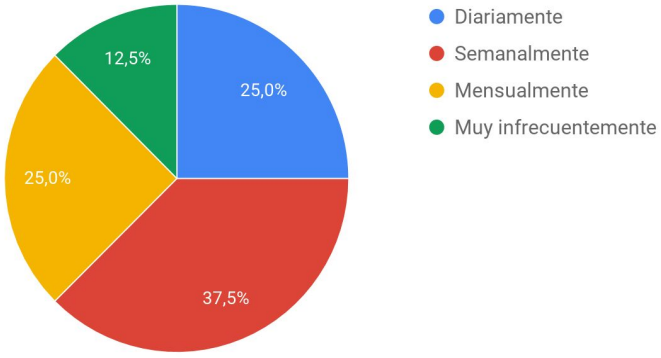
**33**

**15**

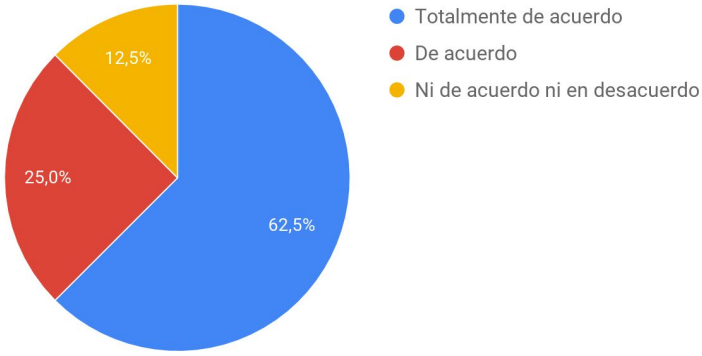
2. ¿Qué dispositivo de realidad virtual tienes?



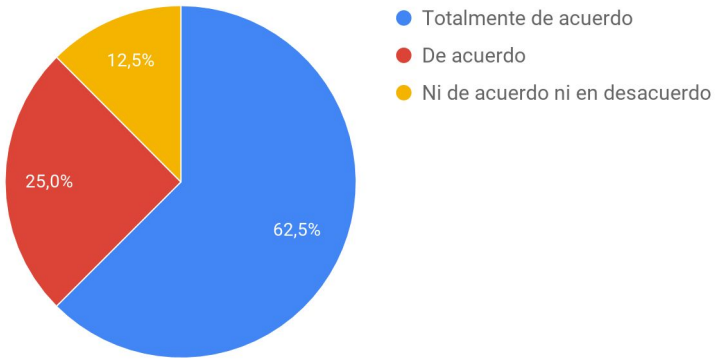
3. ¿Con qué frecuencia sueles jugar a videojuegos de realidad virtual?



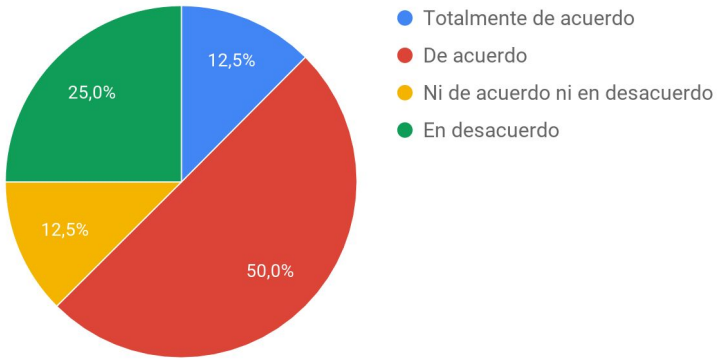

4. En los videojuegos de realidad virtual que he jugado/juego suelo sentir que estoy presente en ese mundo.



5. Considero que los juegos de realidad virtual hacen buen uso de la interacción con el entorno virtual y sus objetos.



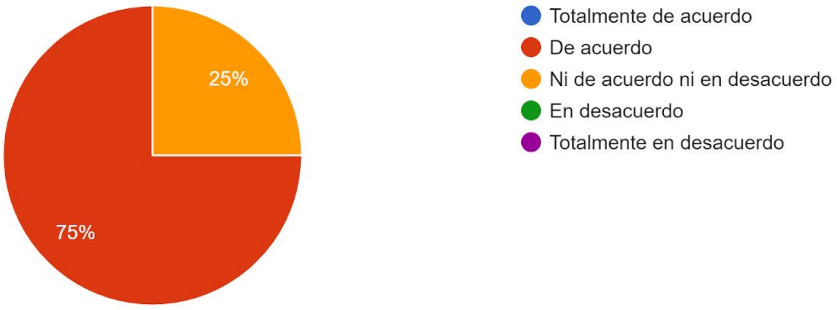

6. Considero que los juegos de realidad virtual hacen buen uso de la interacción con los personajes no jugadores.



## Preguntas posteriores a la prueba

1. Me ha gustado el juego.

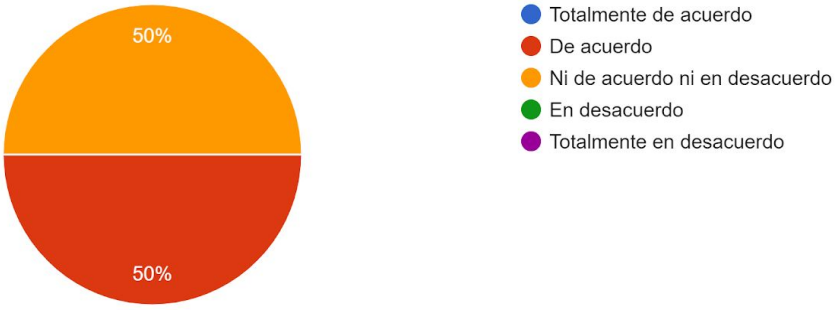
4 respuestas




**Pregunta 1. Test A**

1. Me ha gustado el juego.

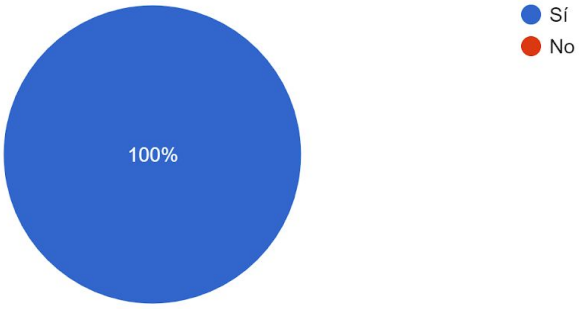
4 respuestas




**Pregunta 1. Test B**

2. He conseguido completar el juego (salir de la habitación).

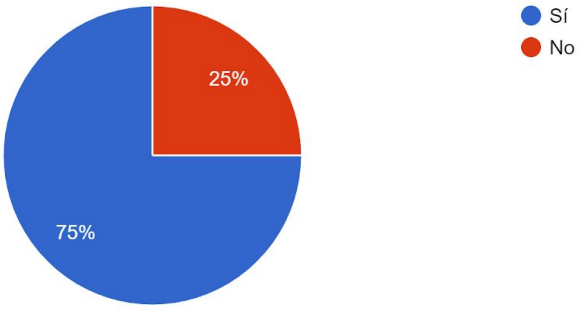
4 respuestas




**Pregunta 2. Test A**

2. He conseguido completar el juego (salir de la habitación).

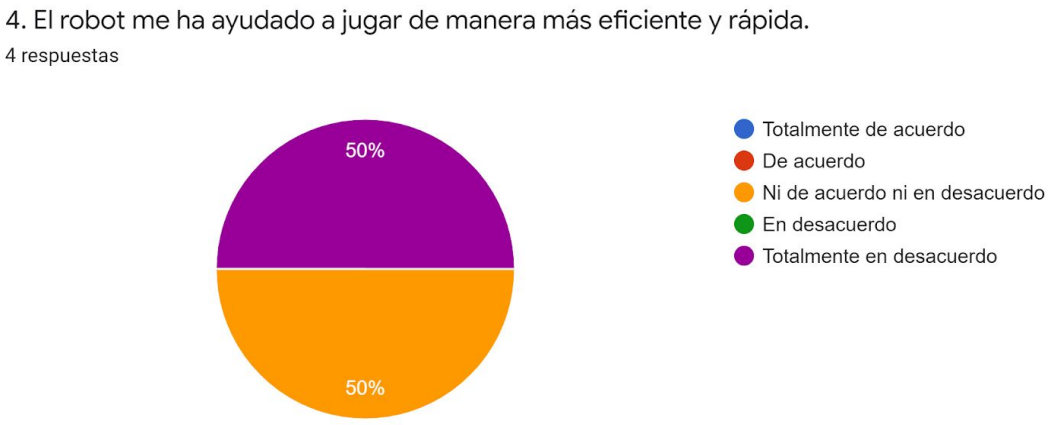
4 respuestas




**Pregunta 2. Test B**

3. En caso de haber completado el juego, ¿cuánto tiempo te ha llevado? Introduce los minutos que ha durado la ejecución, dato que puedes leer en el fichero TiempoJugado.txt que encontrarás en la siguiente ruta de tu ordenador: Equipo > Documentos

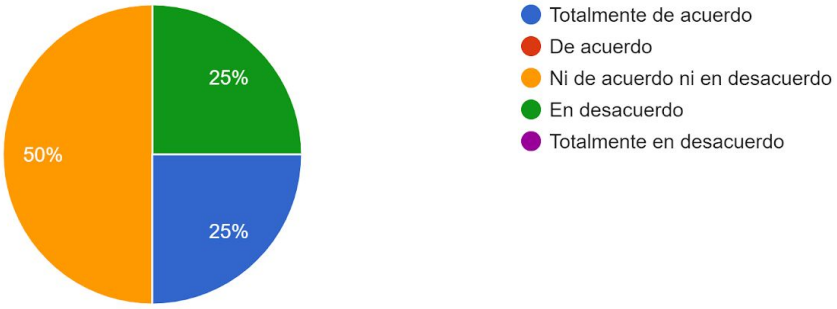
```
24
4.5
4
22
25
9
14
6
```




**Pregunta 4. Test A**

4. El robot me ha ayudado a jugar de manera más eficiente y rápida.

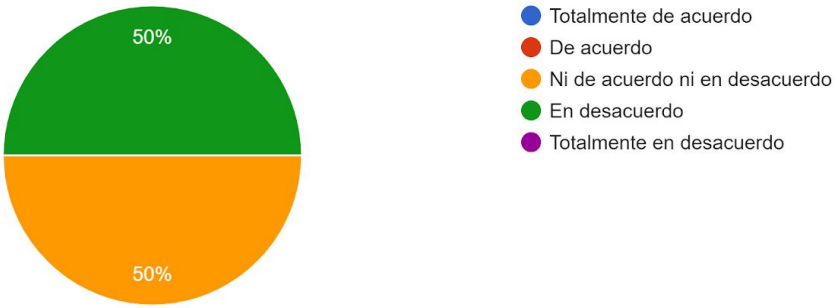
4 respuestas




**Pregunta 4. Test B**

5. El robot se comportaba de manera inteligente.

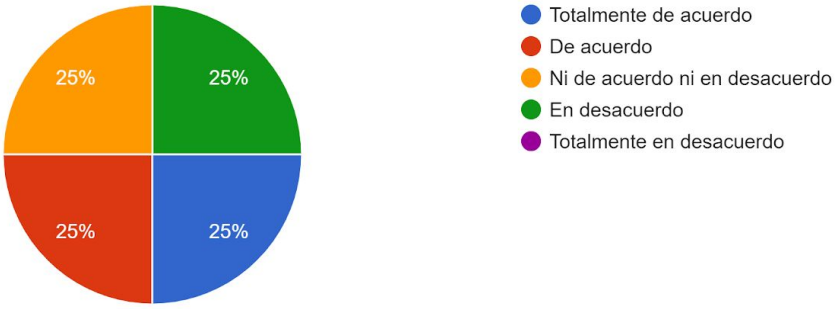
4 respuestas




**Pregunta 5. Test A**

5. El robot se comportaba de manera inteligente.

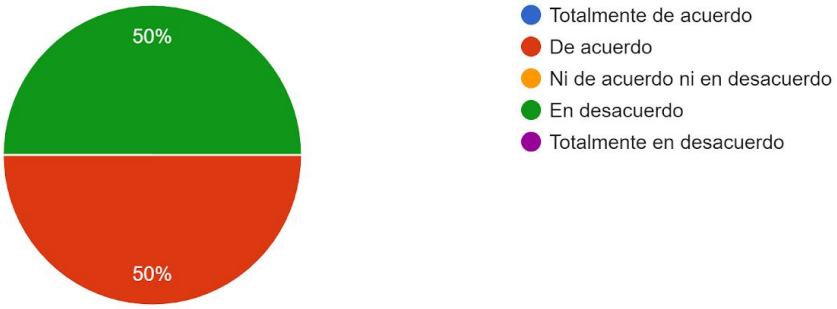
4 respuestas




**Pregunta 5. Test B**

6. He sentido que el robot estaba pendiente de mí y de lo que iba haciendo.

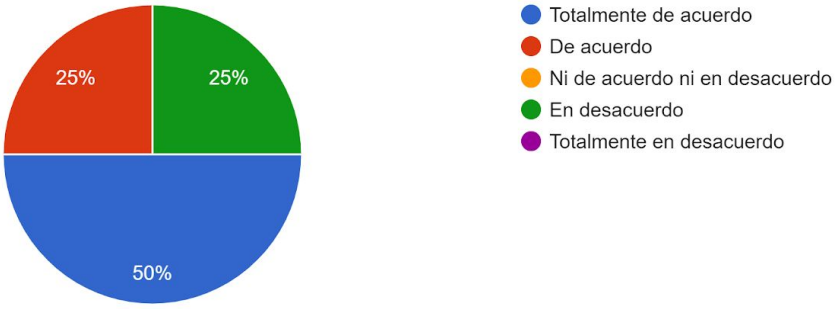
4 respuestas




**Pregunta 6. Test A**

6. He sentido que el robot estaba pendiente de mí y de lo que iba haciendo.

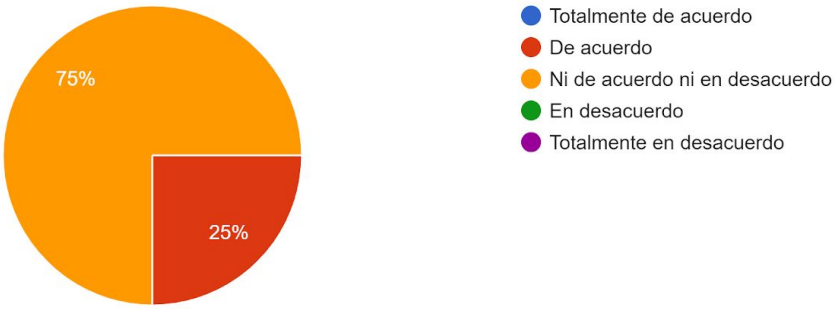
4 respuestas




**Pregunta 6. Test B**

7. He sentido que el robot reaccionaba a mis acciones.

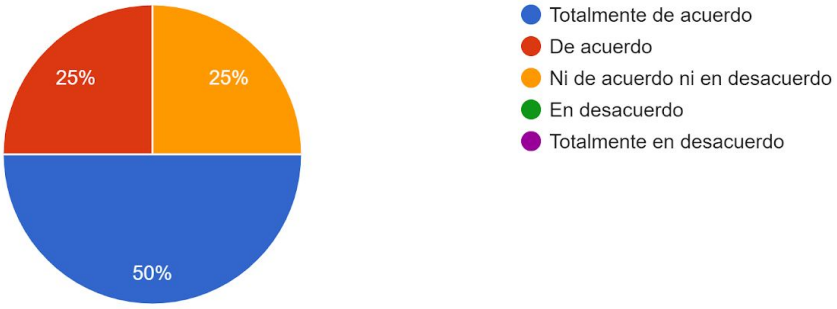
4 respuestas




**Pregunta 7. Test A**

7. He sentido que el robot reaccionaba a mis acciones.

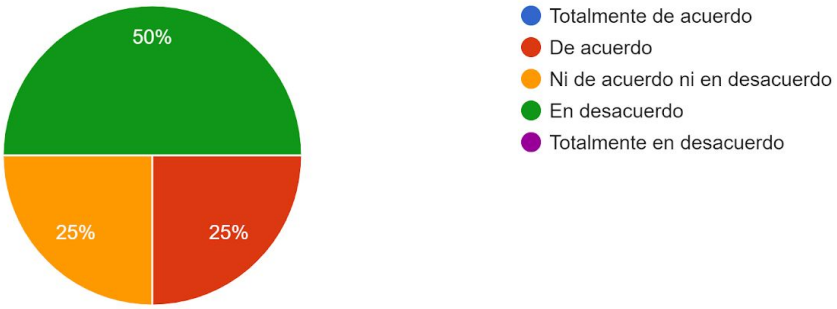
4 respuestas




**Pregunta 7. Test B**

8. He sentido al robot como un verdadero compañero.

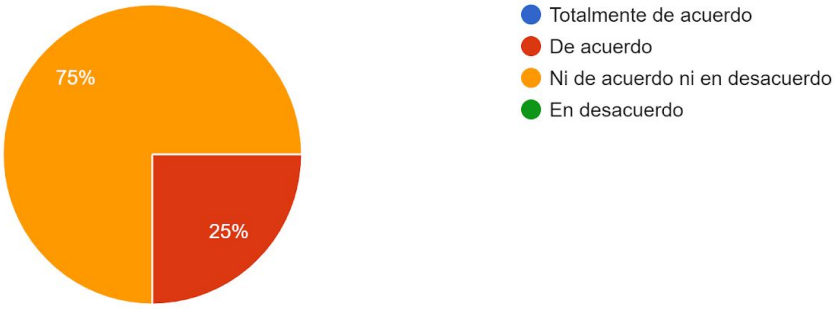
4 respuestas




**Pregunta 8. Test A**

8. He sentido al robot como un verdadero compañero.

4 respuestas




**Pregunta 8. Test B**

9. ¿Ha habido algún momento concreto en el que te hayas sentido perdido? Si es así, cuéntanos cual.

**R1: A la hora de saber dónde tenía que utilizar la poción**

**R2: He empezado moviéndome hacia el tanque de preparado. y tras examinar la mesa con los ingredientes y demás no sabía qué mas hacer hasta que me he dado cuenta de que tenía el otro lado de la sala por explorar.**

**R3: No**

**R4: Al comienzo me ha costado familiarizarme con los controles, debido a que pensaba que no podía teletransportarme a lo largo de la habitación, antes de aparecer los círculos verdes en el suelo.**

**R1: justo al principio no entendía el primer paso, pero en cuanto el robot lo describio todo fue fácil**

**R2: despues de hacer la pocion naranja y echarla en el embudo de la puerta, el liquido lo atravesaba despues de varios botes naranjas descubri que habia q rociar toda la cerradura y no el embudo que seria lo mas intuitivo.**

**R3: The problem with VR is that you aren't sure what is allowed and not allowed, thus examples are needed in order to understand the "rules" of the world. While the hints were helpful, the game didn't**

**seem to have enough cohesion to equip the player with the tools needed to progressively meet the challenges of the game.**

**R4: Se me ha caído cuatro veces la poción naranja**

10. ¿Ha habido algún momento que te haya llamado especialmente la atención? Si es así, explícanos cual.

**R1: El hecho de no poder moverme hasta realizar una serie de acciones lo complica bastante ya que si no se trackea bien el origen del jugador, este puede no ser capaz de llegar a abrir la mesa**

**R2: ninguno en particular**

**R3: El shader de los liquidos de la botella esta muy bien**

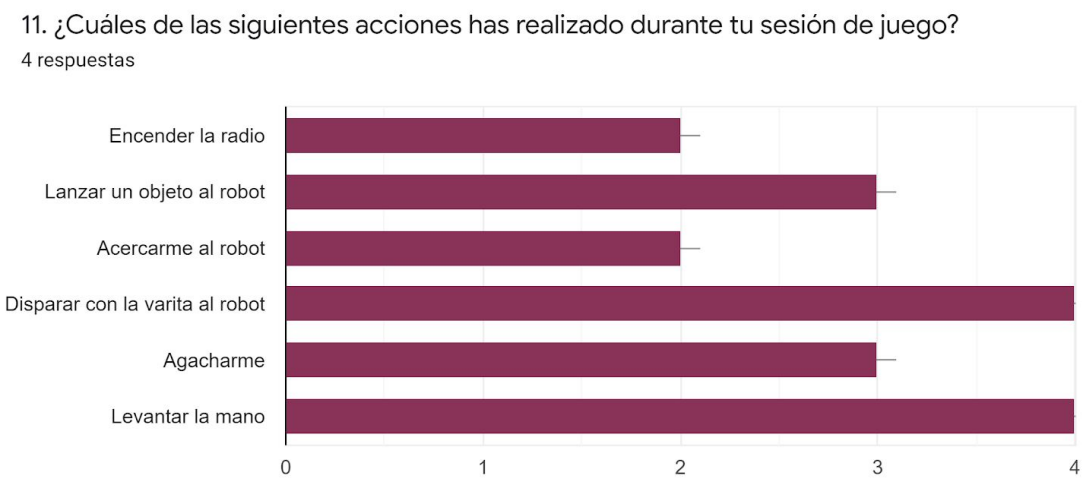
**R4: Me han gustado especialmente los shaders y efectos de simulación de líquidos, y las interacciones de los objetos con la chimenea.**

**R1: Nada a destacar, solo que he echado en falta una forma de recentrar el espacio de juego, dado que he jugado en un espacio reducido me costaba alcanzar los controles del caldero**

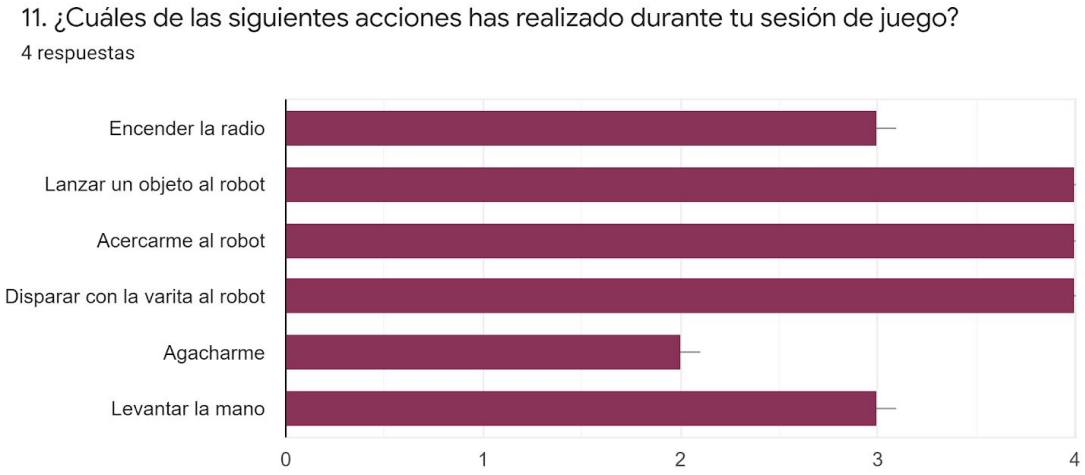
**R2: la varita mágica y el fantasma.. un minijuego curioso.**

**R3: The most entertaining part of the game was actually the radio for me. I'm not sure why. Everything else was interesting. I enjoyed the graphics. The one thing that really caught my attention was the use of the fire in the lantern. Very well done and it looked believable for the setting. Overall the graphics were great for the style of the game.**

**R4: -**




**Pregunta 11. Test A**

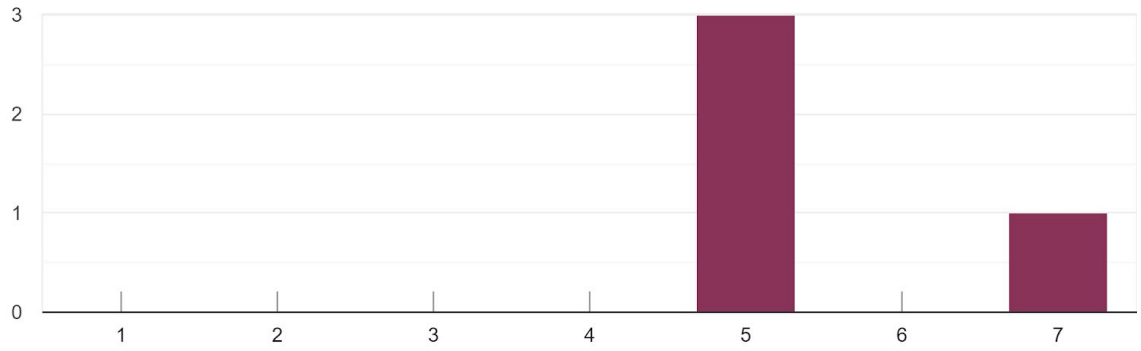



**Pregunta 11. Test B**

## Preguntas referentes a la sensación de presencia

1. Por favor, puntúa tu sensación de "estar dentro" del entorno virtual, en una escala del 1 al 7, donde 7 representa tu sensación natural de encontrarte en un lugar físico real.

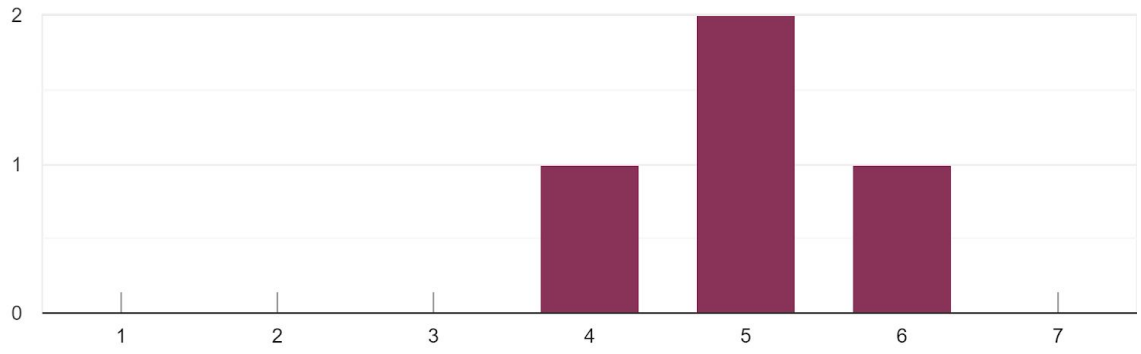
4 respuestas




**Pregunta 1. Test A**

1. Por favor, puntúa tu sensación de "estar dentro" del entorno virtual, en una escala del 1 al 7, donde 7 representa tu sensación natural de encontrarte en un lugar físico real.

4 respuestas




**Pregunta 1. Test B**

2. ¿Hasta qué punto ha habido ocasiones durante la experiencia en las que el entorno virtual era la realidad para ti?

**R1: Soy un jugador demasiado habitual de realidad virtual por lo que esta sensación ya no me ocurre al ser capaz de distinguir la realidad del juego**

**R2: Bastante**

**R3: Al principio, la parte de la llave y la mesa**

**R4: La mayor parte del tiempo**

**R1: ningún momento, como he dicho echaba de falta poder recentrar el juego. me he chocado con muchas cosas "reales" mientras intentaba alcanzar objetos**

**R2: en los momentos estaticos de interaccion.. usando la olla por ejemplo.. el teletransporte resta inmersion quizas si hubiera tenido libre movimiento en el espacio seria inmersion total**

**R3: The interaction with the radio. Although there does need to be collision added to objects vs objects. Like the book not stopping when it hits the shelf (while in hand). It goes right through the shelf, etc.**

**R4: 1**

3. Cuando ahora piensas de nuevo en la experiencia, ¿piensas en el entorno virtual más como una serie de imágenes que has visto o más bien como un sitio en el que has estado?

**R1: Sitio en el que he estado**

**R2: Un sitio en el que he estado**

**R3: Como sitio**

**R4: Como un sitio en el que he estado**

**R1: Como una serie de imágenes**

**R2: imagenes... para sentir presencia tienes que sumergirte mas de 30min.. y movimiento libre para q tu cerebro entienda el espacio.**

**R3: As a place I've been to. It's more of a physical experience than it is traditional entertainment.**

**R4: Como una serie de imágenes que he visto.**

4. Durante la experiencia, ¿qué sensación ha sido más fuerte, tu sensación de encontrarte en un entorno virtual, o de encontrarte en otro lugar?

**R1: Entorno virtual**

**R2: La de encontrarme en otro lugar**

**R3: Entorno virtual**

**R4: Un entorno virtual pero por las limitaciones implícitas del medio, no por la demo en sí. Mi mente es siempre consciente de que no estoy en un entorno físico al no poder chocar con las paredes o mesas. Pero la sensación de presencia está bien conseguida.**

**R1: la sensación de encontrarme en un entorno virtual**

**R2: siendo cartoon es mas virtual.. en otro lugar seria mas en alyx jeje**

**R3: First it was being in another place, then when I was trying to figure out the mechanics, it was finding myself in a virtual environment. The experience of being in another place went away after that due to the "friction" of trying to interact and solve the game.**

**R4: Cuando me he intentado apoyar en una mesa**

5. Haz memoria sobre el momento en el que te has encontrado en el entorno virtual. ¿Cómo de similar ha sido esta experiencia a la "estructura del recuerdo" de otros lugares físicos en los que has estado hoy?

**R1: No llegué a tener esta sensación debido a un par de factores que comentaré en las sugerencias**

**R2: Muy similar**

**R3: No mucho**

**R4: La escala de la habitación y los objetos en general parecen acordes a los elementos del entorno real. Es posible que algunos tal vez sean un poco más grandes de lo que esperaría en relación a mi estatura (1,68), como la mesa inicial, la puerta, o la comida que puedes coger, pero en general lo veo bien.**

**Al recordar el entorno virtual, siento que tengo constancia de las distancias y orientaciones entre la mesa inicial y la máquina de pociones o la puerta, y podría compararlas respecto a distancias del**

**mundo real. Mi mapa mental del entorno virtual no difiere respecto a como percibe mi mente una habitación de la vida real.**

**R1: Pregunta bastante compleja en su formulación y no se si la he entendido bien. Creo que no, que no es un recuerdo para nada similar al de otros lugares físicos donde he estado hoy, normalmente no mido 2,5 metros de altura (la altura de mi setup esta calibrada correctamente, hice double check)**

**R2: la calidad gráfica y ambientación del entorno así como la iluminación estaba bastante conseguida.. si que me ha transmitido presencia y realismo cartoon. pero no comprable con una percepción de espacio físico real**

**R3: I noticed the physical environment was scaled higher than reality, so while the environment felt familiar, it was clear to me that the virtual world was a fantasy based world.**

**R4: -**

6. Durante la experiencia, ¿has pensado de manera frecuente que te encontrabas realmente en el entorno virtual?

**R1: Si, igual que en la pregunta 2, ya que estoy demasiado acostumbrado**

**R2: Cada vez que veía un cartel con instrucciones, de no ser por ello nada.**

**R3: Si**

**R4: Sí**

**R1: Sí constantemente**

**R2: interactuando con los objetos si,**

**R3: In a 'virtual' environment, yes. Not a 'real' environment.**

**R4: No**

2. En este apartado puedes hacer cualquier comentario, sugerencia o aclaración que quieras. Tu opinión nos ayudará a seguir mejorando.

**R1: A la hora de interactuar con objetos como rotaciones o agarrar objetos (supongo que está realizado con SteamVR en Unity), con Skeleton poser generas mejores sensaciones de agarre e involucran mucho más al jugador dentro de la acción en lugar de hacerle desaparecer la mano.**

**A la hora de interactuar con el botón del caldero, este muchas veces (casi todas) no funciona y es necesario utilizar otro objeto para realizar la acción. En el proyecto que estoy realizando he tenido**

**exactamente el mismo problema y es que SteamVR desactiva la colisión de las manos del jugador tras agarrar por primera vez un objeto. Si necesitáis corregir este comportamiento (que es bastante puñetero) no dudéis en contactar conmigo a través del e-mail proporcionado.**

**R2: En mi primera sesión de juego de 22 minutos, aun habiendo preparado la pócima naranja, tuve problemas para verterla en el embudo de la puerta. Si vertía la botella muy cerca del embudo, podía llegar a vaciar la botella entera y no activar el evento de la apertura de puerta, por lo que pensé que necesitaba hacer algo más. Luego me di cuenta de que si lo vertía desde cierta distancia, ya era posible hacerlo funcionar y el evento se activaba al instante.**

**Posteriormente he seguido repitiendo la demo varias veces para probar más cosas.**

**- He observado que es posible teletransportarse al círculo del exterior desde el círculo de la puerta estando cerrada. Apuntando a la puerta y calculando la posición del círculo exterior.**

**- Si creamos dos pócimas grises con X pulsando varias veces el botón de BREW sin verter los objetos correctos, si cogemos una de estas botellas y sin soltarla con la otra mano volvemos a pulsar el botón, como sólo pueden generarse dos botellas, la que tenemos agarrada intenta reaparecer, pero ambos eventos hacen conflicto y desaparece la mano que la agarraba.**

**- Si nos desplazamos en línea recta a lo largo de nuestro espacio de juego real, rotamos la orientación del personaje 180º con el stick y retrocedemos sobre nuestros pasos, si hacemos esto repetidamente es posible atravesar las paredes de la habitación.**

**En relación a la interacción con Jammo, en mi experiencia con la demo, aparte de la introducción inicial donde nos saluda y nos indica mediante un bocadillo de texto que debemos salir de la habitación, y el baile del final cuando finalizamos, no he observado mucha interacción por su parte. Sí que es cierto que se va girando y nos sigue en función de nuestra posición, pero por más que he intentado interactuar con él, saludándole, lanzándole objetos, acercándome, etc, sólo he visto que su comportamiento se reducía a repetir una animación predefinida. En ningún momento he observado que haya intentado comunicarse conmigo o me haya señalado cuál es el siguiente objetivo para darme pistas, no sé si es que he hecho algo mal. Las únicas pistas que me han servido son las de los tableros a lo largo de la habitación. Por eso mi puntuación en sus preguntas ha sido generalmente neutral: personalmente no me ha ayudado pero tampoco me ha perjudicado.**

**Respecto a las indicaciones de los controles, puede ser que sea únicamente cosa mía, pero personalmente me costó entender que para recoger los objetos distantes, tenía que posicionar la palma de mi mano en dirección al objeto para que salga la línea perpendicular, y luego ahí pulsar**

**abajo con el stick. Al principio simplemente apuntaba mi mano al objeto sin girarla y presionaba abajo y no entendía por qué no funcionaba.**

**El estilo visual y artístico está muy bien, es agradable y bonito de ver. Buen uso de materiales, shaders y efectos de partículas.**

**En general, ha sido una experiencia interesante y os deseo lo mejor en futuros proyectos.**

**R3: Personalmente me gusta elegir el "grado de libertad de movimiento" en las aplicaciones de VR. Trabajo desarrollando VR, juego bastante cada día a juegos y aplicaciones "serias" y agradezco mucho que pueda elegir si me desplazo con teletransporte o con movimiento continuo. Entiendo que es un esfuerzo de desarrollo grande, pero en un entorno virtual que el jugador sea el que establezca las reglas que más cómodas le sean es un valor añadido muy grande. Por lo demás 10/10!**

**R4: En general esta bien, lo que más me ha "costado" ha sido entender que el movimiento es con tele-transportes a lugares específicos, pues al iniciar por primera vez el movimiento no ves ningún punto de tele-transporte a no ser que te gires.**

**R1: en lineas generales esta bastante conseguido artisticamente, el uso de objetos es muy bueno, aunque mejorable por ejemplo la llave inicial debería ser absorbida al acercarla a la cerradura. el laser de atraer objetos deberia iluminar el objeto antes de atraerle aunque que salga de la palma de la mano tipo hololens 2 esta muy bien resuelto.**

**el robot se queda un poco en segundo plano.. quizás debería ser mas pequeño para poder ubicarle mas cerca de tu campo de vision.**

**hay un juego en quest llamado de roomVR.. que sigue la misma linea.. echarle un ojo para coger ideas.**

**R2: Todos los juegos VR que he jugado de alguna forma u otra tienen una forma de hacer un recenter y calcular un offset del área de juego al vuelo. por ejemplo en Dead & Buried o robo Recall, usas los dos joysticks hacia abajo y te vuelve a colocar a la altura del suelo adecuada con respecto tu posición actual. Esto Unity lo hace al iniciar la aplicación, si la inicias desde el headset, todo correcto, si la inicias desde el escritorio con el casco puesto en la cabeza o sobre la mesa y luego te pones el casco, toda la experiencia esta jodida.**

**Esta operación siempre se puede hacer al vuelo y cuando yo desarrollaba para VR en Unity era una simple llamaba a un metodo Recenter() de la API.**

**R3:Just some thoughts on improvement: (1) Add collision to the Pick Up Objects so they do not pass through environment meshes. (2) Robot can feel like it gets 'in the way' sometimes when looking around and wanting to teleport to some other place. Perhaps add a simple AI behavior that moves the robot away from players direct line of sight, and moves it to the side IF it blocks line of sight with a teleportation pad. (3) Robot needs more helpful hints and maybe some more audio queues (4) Adding User Thumbstick locomotion would be much preferred over the teleportation method. While VR was in it's early stages, teleportation made sense. But now that more players have adapted to the locomotion, being able to "walk" in game using Thumbstick movement is much more preferred. Also teleportation has a flaw if the player is in a smaller room scale environment. My VR system setup doesn't have a lot of space, so when I reach for things I may hit a wall in real life, etc. Allowing the player to move freely removes this problem. ---- The graphics are great, the particle effects are great, and the mechanics are great. Great job overall. (5) For a full fledged game, this game would need a narrative/story and also some kind of indication of progression.**

Apéndice G

# Estudio sobre videojuegos de Realidad Virtual

Este documento se realizó para analizar los videojuegos más relevantes para el estado de la cuestión. Se ha tomado como referencia de videojuegos exitosos de la gran industria esta lista comercial de mejores juegos para RV: `https://www.techradar.com/best/the-best-vr-games`

Se ha realizado una tabla analizado los diferentes aspectos que hemos considerado más relevantes para la inmersión, reflejándose cada categoría en columnas dentro de la propia tabla.

| JUEGOS | TIPO DE MOVIMIENTO | AVATAR |
|---|---|---|
| **No Man's Sky** | Continuo - Joystick*, Salto - botón* | Manos |
| **Defector** | Continuo - Joystick* | Todo el cuerpo aunque invisible |
| **SuperHot** | Presencial - No joystick. | Manos |
| **Vader Immortal** | Presencial - No joystick + teleport no fijo | Manos con mando |
| **Space Pirate Trainer** | Presencial - No joystick | Sin manos- Únicamente arma |
| **Fallout VR** | Presencial + Joystick + teleport no fijo (Te puedes mover a cualquier mueble o cosa, no solo al suelo) | Se ven los mandos si no coges armas |
| **Beat Saber** | Presencial - No joystick | Espadas laser |
| **Astro bot. Rescue mission** | Es en tercera persona. Controlas un personaje. | Mando de PS4 |
| **Tetris 3D** | Es un juego que no aprovecha las mecánicas de VR. Es un entorno 3D bonito y ya. Mecánicamente no aporta nada. | |
| **Skyrim VR** | Con x o circulo mueves la camara. Con un boton te mueves hacia delante. Tiene opcion configurable de teletransporte | Armas y mano |
| **Moss** | Es en tercera persona. Controlas un personaje. | Mando y manos |
| **Elite dangeours** | Hay movimiento continuo. Nave espacial | |
| **Keep talking and nobody explotes** | Presencial - No joystick. | Mando(Da la sensación de que falta una mano) |
| **Rick morty** | Presencial - No joystick. | Manos |
| **Resident evil 7 VR** | Continuo - Joystick* | Mando y sentado |
| **Batman** | Teletrransporte fijo | Manos y sentado |
| **Arizona Sunshine** | Teletransporte | Manos |
| **Minecraft VR** | Movimiento joystick | Manos de minecraft. No tienen interactvidad |
| **Eve valkyrie** | Movimiento continuo. Estas en una nave espacial. Se mueve en direccion a tu cabeza | Mando |
| **Star Trek: Bridge Crew** | No hay. El personaje está sentado. | Manos |
| **Lucky's Tale** | Es en tercera persona. Controlas un personaje. El movimiento del personaje mueve la cámara | Gamepad. No se ven ni manos ni mandos. |

| INTERFAZ | INTERACCIÓN CON ENTORNO |
|---|---|
| Estática en la escena, Interfaz atada al casco | Interacción realista con palancas y volante |
| Interfaz atada al casco | Interacción realista con botones y escalar |
| Interfaz en la escena | Interacción realista con objetos, se lanzan |
| Interfaz en la escena | Interacción realista con botones y palanca. |
| Interfaz en la escena | Interacción realista con armas |
| Interfaz atada al casco. Manejo de menú con el PAD | Interacción no realista. Se hace con el PAD. |
| Interfaz en la escena | Interacción realista con el sable laser |
| Interfaz en la escena | Manejas a un personaje |
| | |
| Interfaz atada al casco. | Interacción no realista. Se hace con el PAD. |
| Interfaz en la escena | Manejas a un personaje pero te involucras en el entorno. Eres un dios |
| Interfaz en la escena | |
| Interfaz no hay. Es un libro. Si que hay una interfaz | Interacción realista con el maletin |
| No hay interfaz. Interfaz en la escena | Interacción realista |
| Interfaz en la escena | Interacción no realista. Se hace con el mando |
| Interfaz de texto pegado al casco | Interacción realista con las manos |
| Interfaz en la escena. Da indicaciones de mecánicas de VR que no merecen ser explicadas. EJ: Mira a este objeto | Interaccion medio realista con las manos |
| Interfaz en la escena. Atada a la mano izquierda | Interaccion con los objetos equipados. Picar, nadar, atacar etc. |
| Interfaz en la escena. Atada al mando para el tutorial. Interfaz en la nave | Ninguna. Todo es con mando |
| Interfaz en la escena. Interacción con una tablet | Solo se pulsan botones. |
| Interfaz en la escena | Manejas a un personaje |

| TUTORIAL | EXTRAS | MENÚ |
|---|---|---|
| Controles mediante una interfaz de texto | Parece un port directo de la versión original | |
| Controles se muestran en una interfaz de los mandos con texto. Objetivos en una interfaz atada al casco | | 3D. Selección de opcion con puntero laser |
| Tutorial con texto en la escena | | 3D. Menú inicial en una habitación |
| Tutorial en los mandos | | |
| Tutorial en una escena aparte | Salida del guardián con un shader guapo | 3D. Selección de opción con puntero láser |
| Controles se muestran en una interfaz de los mandos con texto. | Para evitar mareos te cierra el campo de visión. Es horrible | |
| Tutorial en escena aparte con texto estático | | 3D. Selección de opción con puntero láser |
| Mediante texto estático | Es raro. No eres tú. Es como si pilotaras un dron | 3D, Selección con pad y botón |
| | | |
| Controles se muestran en una interfaz de los mandos con texto. | Parece un port directo de la versión original | Menú 2D |
| Imagen inicial de controles | Narrador cuento | Menú 3d inicial en una habitación |
| | No entendemos el juego | |
| Tutorial en un cartel en la escena | | Menú 3D inicial |
| | Buenísimo ejemplo | Menú 3D inicial |
| Tutorial inicial con texto pegado a la cámara | | |
| Tutorial inicial con texto en interfaz estática | | Menú 2D inicial. |
| Durante la explicación de los controles se cambian las manos por el mando | Poco realista al introducir texto del juego en la escena. No tiene ningun sentido ver un numero 60 en las balas | Menu 3D Interactuable en una habitación. Se selecciona opciones del menú con un laser desde el mando |
| No hay | Es un port directo | Menu 2D con laser |
| Tutorial en una imagen inicial con los controles de mando | Port directo. | Menu 2D . |
| No hay | Multijugador. Se toman decisiones | No parece que haya |
| Tutorial en carteles | Es un juego normal, pero al que le han añadido las gafas VR, pero no tienen una utilidad real. | No parece que haya |

| COGER OBJETOS | NPC |
| --- | --- |
| | No notan la presencia. |
| No tiene un highlight para mostrar cuándo interactuar. Muchos objetos con los que no puedes interactuar. Se rompen al tirar algunos. Guardas objetos en la interfaz (inventario) | No notan la presencia. Les tocas y es como un puñetazo, poco realista |
| No tiene highlight pero los objetos interactuables se diferencian mucho de los que no. Son Negros | Te atacan, les das golpeas o disparas y mueren, no tienen más interacción que esa porque son básicos |
| No tiene highlitght. Los objetos interactuables se distinguen | NPC Programado, no realista |
| | NPCs básico. Solo te atacan y mueren |
| Coges objetos con botón | NPCs básicos. Te miran |
| No se cogen | No hay |
| No se cogen | No hay |
| | No hay |
| Coges objetos con botón | NPCs básicos. Te miran |
| | |
| | |
| Coger highlight | No hay |
| Coger highlight | NPC no interactivo |
| | NPC no interactivo. Cinemático |
| No tiene highlight al coger cosas | NPC no interactivo. Cinemático |
| Coger . No tiene highlight (tiene un circulo solo en determinados objetros)y no se reconoce bien los objetos que se pueden utilizar. Los objetos los puedes coger desde lejos. | NPCs básicos. Te siguen. Reaccionan a la fisica de las balas |
| No se coge nada | NPCs básicos. Exactamente igual que en minecraft |
| No se coge nada | No hay |
| No se coge nada | NPC no interactivo, Cinemático |
| No se coge nada | No hay |